



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**EFFICIENT XML INTERCHANGE (EXI)
COMPRESSION AND PERFORMANCE BENEFITS:
DEVELOPMENT, IMPLEMENTATION
AND EVALUATION**

by

Sheldon L. Snyder

March 2010

Thesis Advisor:

Don Brutzman

Thesis Co-Advisor:

Don McGregor

**This thesis was done at the MOVES Institute
Approved for public release; distribution is unlimited**

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE March 2010	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE Efficient XML Interchange (EXI) Compression and Performance Benefits: Development, Implementation and Evaluation			5. FUNDING NUMBERS	
6. AUTHOR Sheldon L. Snyder				
7. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT <p>The Department of Defense (DoD) Network-Centric data sharing strategy for the Global Information Grid (GIG) is to XMLize all data. The goal of this strategy is to ensure all data is visible, usable and interoperable, when and where needed, to accelerate decision cycles. However, this XML-based data approach comes at the cost of limiting real-time network edge device connectivity to the GIG because they are seldom able to meet the necessary bandwidth and processing requirements due to XML's intrinsic nature of being verbose and often complex to process.</p> <p>This research explores a powerful and robust solution to XML's network depth limits by means of the World Wide Web Consortium's (W3C) proposed alternative XML format, Efficient XML Interchange (EXI). The EXI format removes redundant tags and values from XML documents and encodes numeric content in a binary format. This format delivers significant file size savings and processing efficiencies compared to existing practices. The evolution of XML's path to EXI is summarized based on the results of the XML Binary Characterization (XBC) working group and the W3C's design points of XML. Followed are recommended steps for EXI development and enterprise integration, focusing on a public open source licensing philosophy. EXI algorithms are described with detailed explanations, Java code samples, and part-task test XML documents. Experiments are conducted evaluating the effectiveness of EXI for DoD tactical use and is followed with a recommended optimal EXI configuration. Several predictive models of EXI's performance are presented to enable potential EXI adopters a measurement tool of expected EXI benefit for various XML domains.</p> <p>This research concludes that for XML-based data, a doubling of bandwidth potential is achievable and CPU burdens minimized when EXI is applied. Additional findings indicate that traditional binary data formats converted to an XML format can be smaller than their native binary format after the application of EXI. Ultimately, through EXI, DoD network edge devices can join the GIG in real-time data exchanges without network hardware refactoring, delivering a more able force.</p>				
14. SUBJECT TERMS Extensible Markup Language (XML), Efficient XML Interchange (EXI), Compression			15. NUMBER OF PAGES 389	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

**EFFICIENT XML INTERCHANGE (EXI) COMPRESSION AND
PERFORMANCE BENEFITS: DEVELOPMENT, IMPLEMENTATION AND
EVALUATION**

Sheldon L. Snyder
Lieutenant, United States Navy
B.S., University of North Florida, 2002

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN MODELING, VIRTUAL ENVIRONMENTS
AND SIMULATION (MOVES)**

from the

**NAVAL POSTGRADUATE SCHOOL
March 2010**

Author: Sheldon L. Snyder

Approved by: Don Brutzman
Thesis Advisor

Don McGregor
Co-Advisor

Mathias Kolsch
Chair, MOVES Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Department of Defense (DoD) Network-Centric data sharing strategy for the Global Information Grid (GIG) is to XMLize all data. The goal of this strategy is to ensure all data is visible, usable and interoperable, when and where needed, to accelerate decision cycles. However, this XML-based data approach comes at the cost of limiting real-time network edge device connectivity to the GIG because they are seldom able to meet the necessary bandwidth and processing requirements due to XML's intrinsic nature of being verbose and often complex to process.

This research explores a powerful and robust solution to XML's network depth limits by means of the World Wide Web Consortium's (W3C) proposed alternative XML format, Efficient XML Interchange (EXI). The EXI format removes redundant tags and values from XML documents and encodes numeric content in a binary format. This format delivers significant file size savings and processing efficiencies compared to existing practices. The evolution of XML's path to EXI is summarized based on the results of the XML Binary Characterization (XBC) working group and the W3C's design points of XML. Followed are recommended steps for EXI development and enterprise integration, focusing on a public open source licensing philosophy. EXI algorithms are described with detailed explanations, Java code samples, and part-task test XML documents. Experiments are conducted evaluating the effectiveness of EXI for DoD tactical use and is followed with a recommended optimal EXI configuration. Several predictive models of EXI's performance are presented to enable potential EXI adopters a measurement tool of expected EXI benefit for various XML domains.

This research concludes that for XML-based data, a doubling of bandwidth potential is achievable and CPU burdens minimized when EXI is applied. Additional findings indicate that traditional binary data formats converted to an XML format can be smaller than their native binary format after the application of EXI. Ultimately, through EXI, DoD network edge devices can join the GIG in real-time data exchanges without network hardware refactoring, delivering a more able force.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	PROBLEM STATEMENT	1
B.	MOTIVATION	3
	1. Extend XML to Unsupported Domains Cases	3
	2. Mitigate XML's Weaknesses	3
	3. Tactical Bandwidth and Small-device Improvements.....	3
C.	2-MINUTE ELEVATOR SPEECH ABOUT THESIS.....	4
D.	RESEARCH QUESTIONS.....	5
	1. Can the Department of Defense (DoD) Keep Up with the Business World's Constantly Connected Internet Applications Philosophy?.....	5
	2. Can DoD Data, using the Extensible Markup Language (XML), be Efficiently Compressed to a Level that Makes Porting to and from Low-bandwidth Military Units Feasible?.....	6
	3. What are the Risks to the DoD Infrastructure if Methods to Push Data Further Down the Echelon Chain of Command are Not Developed?.....	6
	4. Is XML an Effective Tool for Commands that are Under Extremely Low-Bandwidth Constraints?	6
E.	SCOPE OF THESIS	7
F.	METHODOLOGY	8
	1. Background Study	8
	2. Alternative XML Formats	8
	3. Efficient XML Interchange (EXI) Deployment	8
	4. Efficient XML Interchange (EXI) Development.....	8
	5. Statistical Comparison and Modeling of EXI Performance	8
	6. Conclusion	9
G.	THESIS ORGANIZATION.....	9
II.	BACKGROUND AND RELATED WORK.....	11
A.	INTRODUCTION.....	11
B.	WORLD WIDE WEB CONSORTIUM (W3C)	11
C.	XML IN 10 POINTS.....	11
	1. XML is for Structuring Data	11
	2. XML Looks a Bit Like HTML.....	11
	3. XML is Text, but is Not Meant to be Read.....	12
	4. XML is Verbose by Design.....	12
	5. XML is a Family of Technologies.....	12
	6. XML is New, but Not That New	12
	7. XML Leads HTML to XHTML	13
	8. XML is Modular	13
	9. XML is the Basis for RDF and the Semantic Web	13

	10. XML is License-Free, Platform-independent and Well Supported.....	14
D.	DEFINITIONS	14
	1. Focal XML Points within this Thesis	14
	2. Handheld, Mobile and Micro Devices.....	14
	3. Compression	15
	4. Encryption and Digital Signature.....	15
	5. Packaging and Archiving	16
	6. Text-based Compression Generalized.....	16
E.	RELATED WORK	17
	1. GNU Zip (GZip).....	17
	2. Zip.....	18
	3. 7-Zip	20
	4. Efficient XML Interchange (EXI)	21
	5. XMill.....	21
	6. Tar	23
F.	CHAPTER SUMMARY.....	23
III.	XML RELEVANCE	25
A.	INTRODUCTION.....	25
B.	GENERAL XML FORMAT OVERVIEW	25
C.	BREADTH AND DEPTH OF XML	26
	1. Business World XML-Specifics	27
	2. DoD XML Mandates Specifics.....	28
D.	TACTICAL XML RELATED WORK.....	31
	1. Data Interoperability via XSLT Conversions	31
	2. NPS Tract Data Conversion Suite (TDCS).....	33
	3. Semantic Web.....	36
	4. Network-Centric / Force-Net / Future Vision	38
	5. Network Security Considerations.....	39
	6. Tactical Chat	40
	7. Modeling and Simulation (M&S)	41
	8. Internet and Communications Security.....	43
E.	CHAPTER CONCLUSION.....	45
F.	CHAPTER SUMMARY.....	45
IV.	SHORTFALLS OF XML LEADING TO EXI	47
A.	INTRODUCTION.....	47
B.	FILE SIZE	47
	1. XML is Verbose.....	47
	2. Why XML is Verbose	48
	3. XML is Verbose by Design and is Not New	49
C.	PROCESSOR INTENSIVE	51
	1. Legacy File-structure Formatted Data	51
	2. XML's Resolution to the File-Structure Problems	52
	3. Processor Intensity String-to-Number Conversion	52
	4. Processor Intensive Searching	53

D.	IMPACT ON HANDHELD DEVICES	54
1.	Bandwidth Limitations	54
2.	Battery and Heat Limitations	55
a.	Battery Power	55
b.	Heat Dissipation	56
E.	CHAPTER CONCLUSION	57
F.	CHAPTER SUMMARY	57
V.	BINARY XML FORMAT RATIONALE: XBC	59
A.	INTRODUCTION	59
B.	XML BINARY CHARACTERIZATION (XBC) WORKING GROUP	59
C.	DOD INTERESTS IN AN ALTERNATIVE XML FORMAT	63
D.	BUSINESS INTEREST IN AN ALTERNATIVE XML FORMAT	65
1.	Arguments Supporting a Binary XML Format	65
2.	Arguments against a Binary XML Format	86
E.	SYNOPSIS OF INTEREST FOR AND AGAINST AN ALTERNATIVE XML FORMAT	90
1.	Summary of Arguments Pro Binary XML Format	90
2.	Summary of Arguments Con Binary XML Format	91
F.	CHAPTER CONCLUSION	92
G.	CHAPTER SUMMARY	92
VI.	W3C BINARY XML FORMAT (EXI) DECISION JUSTIFICATION AND FRAMEWORK	93
A.	INTRODUCTION	93
B.	VERIFICATION OF CANDIDATE BINARY XML FORMATS: TESTING FRAMEWORK	93
1.	Testing Framework - Measurements	94
2.	Testing Framework - Test-corpus	95
3.	Testing Framework - Drivers	98
4.	Testing Framework - Candidates	100
a.	X.694 ASN.1 with BER	100
b.	X.694 ASN.1 with PER	101
c.	Xebu	101
d.	Extensible Schema-based Binary Compression (XSBC)	102
e.	Fujitsu XML Data Interchange Format (FXDI)	102
f.	Fast Infoset (FI)	103
g.	Efficient XML Interchange (EXI)	103
h.	X.694 ASN.1 with PER + Fast Infoset	104
i.	Efficiency Structured XML (esXML)	104
5.	Testing Framework - Results for Compactness	105
6.	Testing Framework - Results for Processing Efficiency	106
7.	Testing Framework - Results for Round-Trip Conversions	106
C.	EXI SELECTION AND BASELINE (GZIP) TESTING	107
1.	Compactness Comparison	107
2.	Property Comparison	109
3.	Generality Comparison	110

D.	EXI USAGE RECOMMENDATIONS AND LIKELY IMPACT	112
1.	Domain Applicability.....	112
2.	Human Readable.....	112
3.	Domain Optimization	113
4.	Security and Signature	113
a.	Output Alignment.....	113
b.	XML Signature.....	114
c.	XML Encryption	114
5.	Domain Integration.....	115
a.	Web and HTTP Servers	115
b.	XML Modifications Considerations – Schema Mandate	115
c.	Initial EXI Distribution	115
E.	CHAPTER CONCLUSION.....	115
F.	CHAPTER SUMMARY.....	116
VII.	OPENER-EXI IMPLEMENTATION RATIONALE.....	117
A.	INTRODUCTION.....	117
B.	LICENSING	117
1.	Open Source	118
a.	OSS Origins.....	118
b.	OSS Repackaging.....	119
c.	OSS Copy-Left.....	119
d.	OSS Viral.....	119
e.	OSS Examples	120
f.	OSS Conclusion	121
2.	Free Source, Share Source, Shareware.....	121
3.	Proprietary and Commercial.....	122
4.	General Licensing Considerations.....	122
5.	OPENER-EXI Licensing Considerations	123
C.	OPENER-EXI IMPLEMENTATION CONSIDERATIONS.....	124
1.	Web Integration Deployment Focused.....	125
2.	HTTP Negotiated File Format Transfers	125
3.	Apache Server Considerations.....	127
4.	DoD EXI Implementations Considerations	128
a.	EXI Must Become a Recognized Standard	128
b.	EXI Must Not be a DoD-Only Standardized Solution.....	129
c.	Possible Standalone Application	131
d.	EXI and DoD Integration Summary.....	132
D.	APACHE WEB SERVICES IMPLEMENTATION	132
1.	How and Why Apache Is What It Is	132
2.	Establishing an Apache Project.....	134
a.	Vetting Process (Incubation) Overview.....	135
b.	Becoming a Candidate Project (Pre-Podling).....	136
c.	The Podling Phase	137
d.	The Podling Code Release Constraints	139
e.	Graduation into the ASF	140

E.	DOD SYSTEM ACCREDITATION PROCESS	140
1.	Certification and Accreditation (C&A) Process	141
a.	DoD Information Assurance Certification and Accreditation Process (DIACAP)	142
b.	Defense Information Systems Agency (DISA).....	142
c.	System Security Authorization Agreement (SSAA)	143
2.	The Interface Certification Process (ICP)	144
a.	U.S. Navy Afloat ICP Example	146
b.	U.S. Navy Afloat ICR Submission Process	147
c.	U.S. Navy Afloat ICR Endorsement.....	147
d.	U.S. Navy Afloat CM Assumes the ICR for Test and Evaluation (T&E)	148
e.	U.S. Navy Afloat ECP and Installation.....	148
F.	CHAPTER CONCLUSION.....	149
G.	CHAPTER SUMMARY.....	149
VIII.	EXISTRUCTURE AND OPENER-EXI IMPLEMENTATION.....	151
A.	INTRODUCTION.....	151
B.	PREAMBLE.....	151
1.	Source of Reference	151
2.	Chapter Goals.....	151
3.	EXI Introduction.....	152
C.	HEADER	152
1.	Distinguishing Bits and Optional Cookie (Header Part 1 of 5) ...	153
2.	Options Presence Bit (Header Part 2 of 5).....	154
3.	Format Version (Header Part 3 of 5)	154
4.	Options (Header Part 4 of 5).....	156
a.	Alignment Options	159
b.	Strict Option	159
c.	Fragment Option.....	160
d.	Preserve Options	160
e.	Self-contained Option	161
f.	Schema ID Option.....	161
g.	Datatype Representation Map Option	161
h.	Block Size Option	162
i.	Value Max Length Option	162
j.	Value Partition Capacity Option	162
k.	User-Defined Option	163
5.	Padding Bits (Header Part 5 of 5)	163
6.	Graphic User Interface (GUI) Tool for EXI Options	163
D.	EXI BODY	164
1.	String Table	166
a.	Building the String Tables.....	166
b.	Data-structure of the String Table	168
c.	Predefined EXI String Table Initialization Entries.....	169
d.	Default Event Mapping to String Table Entries	173

	<i>e. Local-name String Found</i>	<i>174</i>
	<i>f. String Value Found in Global Only.....</i>	<i>174</i>
	<i>g. String Not Found (Names and Values).....</i>	<i>175</i>
	<i>h. URI String Table Found.....</i>	<i>175</i>
	<i>i. Value Scope.....</i>	<i>176</i>
	<i>j. OPENER-EXI String Table Example.....</i>	<i>179</i>
2.	Grammars and Events.....	181
	<i>a. Information Grammar Theory (Chomsky)</i>	<i>181</i>
	<i>b. Events</i>	<i>182</i>
	<i>c. Event Codes.....</i>	<i>183</i>
	<i>d. Bit and Byte Representation of Events Codes.....</i>	<i>186</i>
	<i>e. Repeating Event and Schema Impact on Event Codes.....</i>	<i>187</i>
	<i>f. Events Implementation Notes and Lessons Learned.....</i>	<i>188</i>
	<i>g. Grammar Creation.....</i>	<i>189</i>
	<i>h. Grammar Document Processing</i>	<i>190</i>
	<i>i. Schema Grammar Building.....</i>	<i>196</i>
	<i>j. Verbose Event and Grammar Encoding of Notebook.xml..</i>	<i>197</i>
	<i>k. Verbose Byte-Aligned Encoding of Notebook.xml Example.....</i>	<i>199</i>
	<i>l. Strict Encoding.....</i>	<i>203</i>
3.	EXI to XML Schema Datatypes and Event/Content Representations.....	203
4.	EXI Datatypes	206
	<i>a. Unsigned Integer Datatype</i>	<i>206</i>
	<i>b. n-bit Unsigned Integer Datatype</i>	<i>209</i>
	<i>c. String Datatype.....</i>	<i>209</i>
	<i>d. Binary Datatype.....</i>	<i>211</i>
	<i>e. Boolean Datatype</i>	<i>211</i>
	<i>f. Float Datatype.....</i>	<i>212</i>
	<i>g. Decimal Datatype</i>	<i>213</i>
	<i>h. Integer Datatype.....</i>	<i>214</i>
	<i>i. QName Datatype</i>	<i>214</i>
	<i>j. Date-Time Datatype</i>	<i>215</i>
	<i>k. Enumerated Datatype</i>	<i>216</i>
	<i>l. List Datatype.....</i>	<i>217</i>
	<i>m. Multiple Ancestor Datatype</i>	<i>217</i>
5.	Datatype Representation Map	217
	<i>a. Primary Alternative Datatype Mapping</i>	<i>217</i>
	<i>b. Secondary Alternative Datatype Mapping</i>	<i>218</i>
	<i>c. Error Reporting for Alternative Datatype Mapping</i>	<i>219</i>
	<i>d. Datatype Mapping Conditional Reporting.....</i>	<i>219</i>
6.	Datatype Compression.....	220
	<i>a. Blocking the Stream of EXI Events into Bins</i>	<i>221</i>
	<i>b. Channelizing the Blocks of Events.....</i>	<i>221</i>
	<i>c. Compressing the Channels into Streams</i>	<i>223</i>

	<i>d. EXI Compression Summary</i>	224
E.	OPENER-EXI XML TEST CASES	225
	1. notebook.xml - Hello World	225
	2. namespace.xml - Namespace Pruning	226
	3. comment.xml - Comment Pruning	227
	4. pi.xml - Processing Instruction (PI) Pruning	227
	5. customer.xml - String Table Values Scope	228
	6. dup.xml - Grammar Transitions on Duplicate Elements/Attributes	229
	7. nestImmediate.xml - Grammar Transitions on Nested Elements with Same Name	229
	8. fullFlex.xml - All Pruning Options	229
F.	SOFTWARE ENGINEERING PRACTICES EMPLOYED	231
	1. Unit Testing	231
	2. Linear Progression	231
	3. Variable Naming	232
	4. Object Oriented Piecewise Methods	232
G.	W3C STATUS OF EXI RECOMMENDATION	233
H.	CHAPTER SUMMARY	234
IX.	DEMONSTRATION AND ANALYSIS OF RESULTS	235
A.	INTRODUCTION	235
B.	DOD-SPECIFIC EXI EXPERIMENTAL TEST CASES DEFINED	235
	1. DoD Modeling and Simulation Sub-Category Test Cases	236
	2. DoD General Sub-Category Test Cases	241
	3. DoD-Only Subcategory Test Cases	243
C.	DOD-SPECIFIC EXI EXPERIMENT RESULTS	243
	1. DoD Modeling and Simulation Test Cases Results	245
	2. DoD General Test Cases Results	249
	3. DoD-Only Test Cases Results	256
	4. DoD EXI Test Cases Summary	261
	5. W3C Corpus of Results	261
D.	RECOMMENDED EXI CONFIGURATION	261
	1. Use of a Schema Whenever Possible	262
	<i>a. N-bit Minimization</i>	262
	<i>b. Datatype Binding</i>	263
	2. No Preservation Options Settings	263
	<i>a. Comments</i>	264
	<i>b. Namespaces</i>	264
	<i>c. Entity and DTD</i>	264
	<i>d. Processing Instructions (PI)</i>	265
	3. Use Post-Processing Compression	266
	4. Experiment of the Recommended EXI Configuration Settings	267
	<i>a. Preservation</i>	267
	<i>b. Alignment</i>	268
	<i>c. XML Schema</i>	268

	d.	<i>Conclusion of Recommended EXI Configuration Experiment</i>	268
E.		TEST-CORPUS FOR STATISTICAL COLLECTIONS	270
F.		EXI COMPRESSION STATISTICAL SIGNIFICANCE TEST	273
	1.	Friedman Non-Parametric Test for Randomized Block Experiments	273
	a.	<i>Friedman Non-Parametric Technique Equations</i>	274
	b.	<i>Friedman Non-Parametric Example Demonstration</i>	276
	c.	<i>EXI Non-Parametric Friedman Analysis</i>	277
	d.	<i>EXI Non-Parametric Multiple Comparison Tukey Method</i>	278
	e.	<i>Conclusion of EXI Non-Parametric Friedman Analysis</i>	279
	2.	Analysis of Variance (ANOVA)	279
	a.	<i>ANOVA Equations</i>	280
	b.	<i>Conclusion of EXI ANOVA</i>	282
	3.	Conclusion of Test of Significance Between Compression Techniques	284
G.		STATISTICAL PREDICTIVE MODELS	285
	1.	Model Factors of Interest	286
	2.	Sampled Data Assumptions and Mitigation for Modeling	287
	a.	<i>Independence of the observation in the sample</i>	287
	b.	<i>Linearity of the expected value as a function of the independent variables</i>	287
	c.	<i>Equal variance of the errors of the dependent variable</i>	288
	d.	<i>Normally distributed dependent variable</i>	288
	e.	<i>Data Transformation Consideration–Dependent Variable</i>	289
	f.	<i>Data Transformation Considerations–Independent Predictor Variables</i>	290
	3.	Variance Between and Within XML Documents	292
	a.	<i>The Source of Variance</i>	292
	b.	<i>Example of Variance</i>	292
	c.	<i>Variance Mitigation Techniques</i>	293
	4.	Parametric Prediction Model	294
	a.	<i>Parametric Models in General</i>	294
	b.	<i>Parametric Model Measure of Variance</i>	295
	c.	<i>Parametric Model Equation</i>	296
	d.	<i>Parametric Model Usage Characteristics</i>	297
	e.	<i>Parametric Model Parameter Interpretation</i>	299
	f.	<i>Parametric Model Factor Profile</i>	302
	g.	<i>Parametric Model Factor Impact on Model</i>	302
	h.	<i>Parametric Model Analysis of Fit and Feasibility</i>	303
	i.	<i>Parametric Model Comparison to Sample Data</i>	307
	j.	<i>Parametric Model Conclusions</i>	309
	5.	Non-Parametric Prediction Model	309
	a.	<i>Non-Parametric Model Design Points</i>	309
	b.	<i>Non-Parametric Model Developed</i>	311

	<i>c.</i>	<i>Non-Parametric Model Results Interpretation</i>	<i>315</i>
	<i>d.</i>	<i>Non-parametric Model Analysis of Fit and Feasibility.....</i>	<i>317</i>
	<i>e.</i>	<i>Non-Parametric Model Comparison to Sample Data.....</i>	<i>319</i>
	<i>f.</i>	<i>Non-Parametric Model Conclusions.....</i>	<i>321</i>
6.		Conclusions Regarding EXI Prediction Models	321
	<i>a.</i>	<i>Significance between the General Models</i>	<i>321</i>
	<i>b.</i>	<i>Domain-Specific Models.....</i>	<i>323</i>
	<i>c.</i>	<i>EXI Models in General.....</i>	<i>324</i>
H.		EXI IMPLEMENTATIONS AND TOOLS.....	325
	1.	Available EXI Implementations	325
	2.	NPS EXI Comparison Tool.....	326
	3.	NPS Options Tool.....	326
I.		CHAPTER CONCLUSION.....	327
J.		CHAPTER SUMMARY.....	328
X.		CONCLUSIONS AND RECOMMENDATIONS.....	329
	A.	CONCLUSIONS	329
		1. The Technology Development and Adoption Litmus Test.....	329
		2. Research Questions Answered.....	330
		<i>a.</i> <i>Can the Department of Defense (DoD) Keep Up with</i> <i>Enterprise America's Constantly Connected Internet</i> <i>Applications Philosophy?</i>	<i>330</i>
		<i>b.</i> <i>Can DoD Data, using the Extensible Markup Language</i> <i>(XML), be Efficiently Compressed to a Level that Makes</i> <i>Porting to and from Low-bandwidth Military Units</i> <i>Feasible?.....</i>	<i>331</i>
		<i>c.</i> <i>What are the Risks to the DoD Infrastructure if they Do</i> <i>Not Develop Methods to Push Data Further Down the</i> <i>Echelon Chain of Command?</i>	<i>331</i>
		<i>d.</i> <i>Is XML an Effective Tool for Commands that are Under</i> <i>Extremely Low-bandwidth Constraints?.....</i>	<i>332</i>
	B.	RECOMMENDATIONS FOR FUTURE WORK.....	332
		1. Full EXI Specification of OPENER-EXI	332
		2. Develop a Micro Version of EXI.....	333
		3. Create an Example of the Motivation Scenario	333
		4. DoD Integration Package	334
		5. Demonstrate the EXI Processing	334
		6. Efficient EXI Fragments in Line with XML	334
		7. Department of the Navy (DON) Needs to Join the W3C.....	336
		LIST OF REFERENCES.....	337
		INITIAL DISTRIBUTION LIST	355

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1.	EXI Enabling Real-Time High and Low-Bandwidth Locations to Interoperate	5
Figure 2.	Notional Step-by-Step Example of Network Throughput Optimizing by Negotiated Compression	18
Figure 3.	Zip Archive Notional File Structuring (From Zip, 2010)	19
Figure 4.	Example 7-Zip File Manager User Interface	20
Figure 5.	Example 7-Zip Output Configuration Settings Interface	21
Figure 6.	Notional XML's Extensible Stylesheet Language Transformations (XSLT) Process flow	32
Figure 7.	C4I Systems with Common Tract Data Interchanges	34
Figure 8.	Tract Data Conversion Suite Conceptual Architecture	35
Figure 9.	Example of Transfer Time vs. Bandwidth (Data Rate) for a 100MB Video or Imagery File (From Popovich, 2005)	54
Figure 10.	Binary XML Format Design Property Demands (From W3C, 2005)	62
Figure 11.	KDDI Research XMark Document Format Compression Comparison (From KDDI, 2003)	67
Figure 12.	Systematic's Result Chart of Technique Comparisons (From Systematic Software, 2003)	71
Figure 13.	SUN's Traditional Web Services Pipeline (From Sun, 2003)	73
Figure 14.	SUN's Recomend Self-Documenting, Length Prefixed XML Encoding (From Sun, 2003)	74
Figure 15.	SUN's Recomend Fast Schema Pipeline (From Sun, 2003)	74
Figure 16.	SUN's Comparison of Results Chart between Traditonal XML and Fast Schama (From Sun, 2003)	75
Figure 17.	Advanced Technologies Group, NDS Satellite Communication Loop Diagram (From Advanced Technologies Group, 2003)	78
Figure 18.	Sosnoski Encode Time Comparison of XBIS and XML (From Sosnoski, 2003)	81
Figure 19.	Sosnoski Decode Time Comparison of XBIS and XML (From Sosnoski, 2003)	81
Figure 20.	CCSDS Space Domain Functional Areas for XML Implementation Considerations (From CSC/NASA, 2003)	83
Figure 21.	W3C Testing Framework Flowchart for Candidate Binary XML Formats (From W3C, 2006)	94
Figure 22.	W3C Binary XML Test-Corpus of Documents by File Size and Value Content Density (From W3C, 2007)	96
Figure 23.	EXI Compactness Comparison to Traditional GZip (From W3C, 2008)	108
Figure 24.	EXI Integration with XML Encryption and Signature (From Williams, 2009)	114
Figure 25.	EXI as a Negotiable Compression Technique at the Web Server	126
Figure 26.	Web Servers' Worldwide Market Share (After Netcraft, 2009)	127
Figure 27.	HLA General Architeture Overview Example	130

Figure 28.	Apache Notional Project Adoption Vetting Flow (From Apache, n.d.)	135
Figure 29.	High Level Overview of the DIACAP Process	144
Figure 30.	Notional ICR Progression Events Waterfall Chart	145
Figure 31.	DoD Afloat Change Request Flowchart Example	146
Figure 32.	OPENER-EXI's Graphic User Interface Capturing All EXI Options and Settings.....	164
Figure 33.	Basic EXI Namespace Driven String Table Design	167
Figure 34.	EXI URI String Table Partition Initial Entries (From W3C, 2008)	170
Figure 35.	EXI Prefix String Table Partition Initial Entries (From W3C, 2008)	171
Figure 36.	Local-Name String Table Entries Based on Notebook.xml Example (From W3C, 2008).....	177
Figure 37.	Global and Local Values String Table Entry Mapping to Local-Name (From W3C, 2008).....	177
Figure 38.	EXI Grammar Learning, Discovery and Transition Processes Based on the notebook.xml Document (From W3C, 2008)	192
Figure 39.	Abstract EXI Grammar Creation Process Diagram	193
Figure 40.	Abstract EXI Grammar Transition Process	194
Figure 41.	Color Coded Overview of notebook.xml Grammar/Event Encoding (From W3C, 2008).....	195
Figure 42.	EXI String Processing Model (From W3C, 2008)	211
Figure 43.	EXI Compressed Aligned Output Events Mapping to Compression Channels (From W3C, 2008).....	222
Figure 44.	EXI Events To Compressed Stream for EXI Compression Aligned Output (From W3C, 2008).....	224
Figure 45.	Open CV HAAR Facial Detection Example Results.....	236
Figure 46.	Example Rendered SVG File (From Inkscape, n.d.)	237
Figure 47.	VISKIT/SIMKIT Example M/M/1 Queue Event Graph	238
Figure 48.	VISKIT/SIMKIT Example M/M/1 Queue Assembly	238
Figure 49.	Example X3D 3D Scene of the 5 Platonic Solids.....	239
Figure 50.	Example Delta3D Humaniod Map.....	239
Figure 51.	Example Delta3D Scene Map.....	240
Figure 52.	Microsoft Word 2007 Baseline File-Structure (From Microsoft, 2006).....	242
Figure 53.	Comparison of EXI Encodings of the DoD M&S Test Case Documents	246
Figure 54.	Comparison of EXI Schema-Informed Encodings of the DoD M&S Test Case Documents.....	247
Figure 55.	Comparison of EXI Encodings Baselined to GZip for the DoD M&S Test Case Documents.....	248
Figure 56.	Comparison of EXI Schema-Informed Encodings Baselined to GZip for the DoD M&S Test Case Documents	249
Figure 57.	Comparison of EXI Encodings on the DoD General Test Case Documents.....	252
Figure 58.	Comparison of EXI Schema-Informed Encodings of the DoD General Test Case Documents.....	253
Figure 59.	Comparison of EXI Encodings Baselined on GZip for the DoD General Test Case Documents.....	254

Figure 60.	Scatter Plot Comparison of EXI Schema-Informed Encodings Baselined on GZip for the DoD General Test Case Documents	255
Figure 61.	Comparison of EXI Encodings of the DoD-specific Test Case Documents	257
Figure 62.	Comparison of EXI Schema-Informed Encodings of the DoD-Specific Test Case Documents.....	258
Figure 63.	Comparison of EXI Encodings Baselined on GZip for the DoD-specific Test Case Documents.....	259
Figure 64.	Comparison of EXI Schema-informed Encodings Baselined on GZip for the DoD-specific Test Case Documents	260
Figure 65.	Example of the Central Limit Theorem (From Sanchez, 2009)	279
Figure 66.	Analysis of Variance (ANOVA) Comparison of XML Techniques	283
Figure 67.	Distribuiton of Compared XML Technique–Percentage of Original Document.....	285
Figure 68.	EXI Restuls Distributions (Transformed).....	290
Figure 69.	Predictor Variables by EXI Schemaless Results (Untransformed).....	291
Figure 70.	Predictor Variables by EXI Schemaless Results (Transformed)	291
Figure 71.	Parametric Model Factor Profiles, How One Unit Change in a Factor Effects the EXI Predicted Results.....	302
Figure 72.	Parametric Model Terms Impact on the Compression Results.....	303
Figure 73.	Parametric Model Actual by Predicted Plot.....	304
Figure 74.	Parametric Model Plot of Residuals by Predicted Value.....	305
Figure 75.	Parametric Model Distribution of Residuals.....	305
Figure 76.	Non-Parametric Model Split History	312
Figure 77.	Non-Parametric Detailed CART Tree	313
Figure 78.	Non-Parametric Simple Branching CART Tree	314
Figure 79.	Non-Parametric CART Factor Effect on Model.....	317
Figure 80.	Non-Parametric CART Model Distribution of Residuals.....	318
Figure 81.	Analysis of Variance (ANOVA) Comparison of Predictive Models.....	322
Figure 82.	Analysis of Variance Between Parametric and Non-Parametric Models for General and Domain-Specific XML Cases.....	323
Figure 83.	Technique Comparison Tool.....	326
Figure 84.	EXI Options Tool.....	327

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1.	Simple XML Document, EXI's Unofficial "Hello World" (From W3C, 2007)	26
Table 2.	Network-centric Notional Community of Interest (COI) Data-Sharing Roles (From DOD CIO IM, 2006).....	39
Table 3.	Notebook.xml Verbose Structured Format (From W3C, 2007)	47
Table 4.	Notebook.xml Terse Information-Only Format.....	48
Table 5.	XML Structure Bloating Example (X3D XML Code Top, X3D Scene Bottom)	50
Table 6.	String-to-Numeric Conversion Algorithm.....	53
Table 7.	W3C List of Domains That are Unsupported by the Native XML 1.x Format (From W3C, 2005)	61
Table 8.	MITRE XML Compression Comparisons Study on DoD Messages (After MITRE, 2008).....	64
Table 9.	Systematic's XML Reformating Strucutre Example (From Systematic Software, 2003).....	70
Table 10.	SUN's Table Result Property Comparison (From Sun, 2003)	75
Table 11.	CubeWerx Comparison (in Kilo Bytes) of Proposed BXML and XML (From CubeWerx, 2003)	77
Table 12.	Advanced Technologies Group, NDS Table of Sampled Alternative XML Format Techniques (From Advanced Technologies Group, 2003)	79
Table 13.	W3C Binary XML Test-Corpus of XML Documents Listed by Use Group (From W3C, 2007).....	97
Table 14.	W3C Binary XML Framework Test for Application Classes for XML Structure (From W3C, 2007)	98
Table 15.	W3C Binary XML Framework Test Results Summary of Percentage of Improvement for Compactness Over Baseline (From W3C, 2007)	106
Table 16.	W3C Binary XML Framework Test Results Summary of Percentage of Improvement for Processing Efficiency Over Baseline (From W3C, 2007).106	
Table 17.	Comparison of W3C Binary XML Property Requirements Between GZip and EXI (From W3C, 2008)	109
Table 18.	Comparison of W3C Binary XML Property Demands Between GZip and EXI (Must Not Prohibit) (After W3C, 2008).....	110
Table 19.	Binary XML Generalized Comparison of EXI and Gzip (From W3C, 2008)	111
Table 20.	Comparison of End-User Rights for Common OSS Licenses (After Laurent, 2004 & Beard, 2007)	120
Table 21.	Developer Perspective Licensing Considerations (After Laurent, 2004 & Michaelson, 2004).....	123
Table 22.	Basic EXI Stream Structure	152
Table 23.	EXI Header Format.....	153
Table 24.	Minimum Bits to Distinguish an EXI Stream from Other Text-Based XML Streams.....	153

Table 25.	EXI Stream Distinguishing Bits with Optional 4-Byte Cookie	153
Table 26.	EXI Header 4-Bit Version Unsigned Integer Examples without Prefixed Version Flag.....	155
Table 27.	EXI Header Version Number Pseudo Code.....	156
Table 28.	EXI Header Version Field Examples.....	156
Table 29.	EXI Header Options and Default Values (From W3C, 2008)	157
Table 30.	Options Schema for EXI Header Options XML Generation (From W3C, 2008)	158
Table 31.	EXI Fidelity Options: Event Preservation Options (From W3C, 2008)	160
Table 32.	EXI Stream Encoding Pseudo Algorithm	165
Table 33.	EXI URI String Table Partition Initial Entries (After W3C, 2008)	170
Table 34.	EXI Prefix String Table Partition Initial Entries (After W3C, 2008)	170
Table 35.	EXI Default XML Namespace String Table Local-Name Entries (From W3C, 2008).....	171
Table 36.	Default XSI Namespace String Table Local-Name Entries (From W3C, 2008)	171
Table 37.	Default XSD Namespace String Table Local-name Entries (From W3C, 2008)	172
Table 38.	EXI Default XML Event Mapping to String Table Partition (From W3C, 2008)	173
Table 39.	OPENER-EXI (Notebook.xml) String Table Build Example Output	179
Table 40.	The notebook.xml Local Copy (From W3C, 2008).....	180
Table 41.	String Table Creation Pseudocode Algorithm	180
Table 42.	EXI Defined Event Types and Notation (From W3C, 2008)	183
Table 43.	EXI Grammars by Events Structure (After W3C, 2008).....	184
Table 44.	Elementary Event Codes Example (From W3C, 2008).....	186
Table 45.	Verbose Comparison of Event Codes between Bit and Byte Alignments (From W3C, 2008).....	187
Table 46.	Rigid Schema Formatting Compliance Requirment for XML Documments Example Case	196
Table 47.	Notebook.xml Local Copy for Quick Reference	202
Table 48.	Pseudocode Algorithm for Decoding EXI Streams	203
Table 49.	Default Datatype Of EXI Events (From W3C, 2008).....	204
Table 50.	Schema to EXI Default Datatype Transformation Mapping (From W3C, 2008)	205
Table 51.	EXI Built-In Datatype Character Restrictions (From W3C, 2008)	206
Table 52.	Verbose EXI Unsigned Integer Value Examples.....	207
Table 53.	EXI Unsigned Integer Decoding Pseudocode Algorithm.....	207
Table 54.	EXI Unsigned Integer Encoding Pseudocode Algorithm	207
Table 55.	EXI Unsigned Integer Decoding Java Method	208
Table 56.	EXI Unsigned Integer Encoding Java Method	208
Table 57.	EXI String Restricted Set Pseudocode Algorithm	210
Table 58.	EXI Float Decoding Pseudocode Algorithm	213
Table 59.	EXI Decimal Encoding Pseudocode Algorithm	214
Table 60.	EXI QName Rules	215

Table 61.	Components Fields of a Date and Time Datatype (From W3C, 2008)	215
Table 62.	Schema To EXI Date and Time Component Mapping (From W3C, 2008) ..	216
Table 63.	Example EXI DatatypeRepresentationMap XML Document (After W3C, 2008)	218
Table 64.	Compression Results Comparison for DoD M&S Test Case Documents	245
Table 65.	Compression Results Comparison for General Use DoD Test Case Documents	250
Table 66.	Compression Results Comparison for DoD-Specific Test Case Documents	256
Table 67.	Rowhead Parameter-Dependent Results of Full-Factorial Compression Experiment.....	267
Table 68.	Recommended EXI Configuration Experiment Test Document HelloWorld.x3d (From Brutzman, 2007)	269
Table 69.	Extensions of the W3C Test-Corpus (After W3C, 2008)	270
Table 70.	Friedman Example Experiment Data set (From DeVore, 2008)	276
Table 71.	Friedman Example Experiment RANKED Data set (From DeVore, 2008) ..	276
Table 72.	Friedman EXI Experiment RANKED Data set Summary	277
Table 73.	EXI Sorted Ranked Averages	278
Table 74.	EXI Block Averages Differences.....	278
Table 75.	ANOVA Calculation Table (After DeVore, 2008).....	282
Table 76.	Moments of the ANOVA Comparison of XML Techniques.....	284
Table 77.	Tukey-Kramer Statistical Significance Difference between Techniques Measurement.....	284
Table 78.	XML Descriptive Factors Captured for EXI Prediction Modeling	286
Table 79.	Transformation Algorithms Attempted on EXI Results	289
Table 80.	XML Document Variance Example (Small)	292
Table 81.	XML Document Variance Example (Large)	293
Table 82.	XML Document Variance Experiment Test Results	293
Table 83.	Summary of Parametric Model Fit (Transformed Data).....	295
Table 84.	Parametric Model Parameter Estimates (Transformed Data)	297
Table 85.	Parametric Model Results Transformation Examples	298
Table 86.	Parametric Model Distribution of Residuals Quartile Range	306
Table 87.	Parametric Model Distribution of Residuals Moments	306
Table 88.	Parametric Model Comparison of Results–Score of Model	308
Table 89.	CART Leaf Summary	316
Table 90.	Non-Parametric CART Model Distribution of Residuals Quartile Range	318
Table 91.	Non-Parametric CART Model Distribution of Residuals Moments.....	319
Table 92.	Non-Parametric Model Comparison of Results.....	320
Table 93.	Domain-Specific X3D Models Comparison to General XML Models Statistics	324

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

3D	Three Dimensional
AAR	After Action Review After Action Report
AI	Artificial Intelligence
AIT	Automated Information Technology
ASCII	American Standard Code for Information Interchange
AOR	Areas of Operations
AOU	Area Of Uncertainty
API	Application Programming Interface
ASD	Assistant Secretary of Defense
ASF	Apache Software Foundation
ASL	Apache Software License
ASM	Anti-Ship Missile
ASN	Abstract Syntax Notation
ATL	Acquisition, Technologies and Logistics
ATO	Authority To Operate
AVCL	Autonomous Vehicle Command Language
BER	Basic Encoding Rules
BSD	Berkeley Software Distribution
BXML	Binary XML
C&A	Certification and Accreditation
CA	Certification Agent
CAD	Computer Aided Drawing
CCSDS	Consultative Committee for Space Data Systems
CIO	Chief Information Officer
COA	Course of Action
COI	Community of Interest
CM	Configuration Manager

CMF	Common Message Format
CNNWC	Commander Naval Network Warfare Command
CNO	Chief of Naval Operations
COI	Community of Interest
COP	Common Operational Picture
CORBA	Common Object Request Broker
COTS	Commercial Off-the-Shelf
CPL	Certified Parts List
DAA	Designated Approval Authority
DAML	DARPA Agent Markup Language
DDMS	DoD Discovery Metadata Specification
DEG	Discrete Event Graph
DES	Discrete Event Simulation
DIACAP	DoD Information Assurance Certification and Accreditation Process
DIS	Distributed Interactive Simulation
DISA	Defense Information Systems Agency
DoD	Department of Defense
DoE	Design of Experiments
DOM	Document Object Model
DON	Department of the Navy
DTD	Document Type Definition
DVR	Digital Video Recorder
ECP	Engineering Change Plan
EOF	End of File
ESG	Electronic Service Guides
esXML	Efficiency Structured XML
EULA	End User License Agreement
EXI	Efficient XML Interchange
FI	Fast Infoset

FSM	Finite State Machine
FXDI	Fujitsu XML Data Interchange
GIG	Global Information Grid
GIGO	Garbage In Garbage Out
GIS	Geographic Information Systems
GML	Geographic Markup Language
GO	Government Organizations
GPL	GNU Public License
GUI	Graphic User Interface
GNU	Gnu's Not Unix
GWOT	Global War On Terrorism
GZip	Gnu's Not Uniz Zip
HBAF	Hybrid Byte-Aligned Format
HHQ	Higher Headquarters
HLA	Higher Level Architecture
HLS	Homeland Security
HPC	High Performance Computing
HTTP	Hyper Text Transfer Protocol
I18N	internationalization
IATO	Interim Authority to Operate
ICP	Interface Certification Process
ICR	Interface Change Request
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
INFOSET	XML Information Set
ISO	International Standards Organization
IT	Information Technology
ITU-T	International Telecommunications Union - Telecommunications
JAR	Java Archive
JIT	Just-In-Time

JVM	Java Virtual Machine
KBC	Knowledge Based Compression
LGPL	GNU Lesser General Public License
LZMA	Lempel-Ziv-Markov Chain Algorithm
KM	Knowledge Management
M&S	Modeling and Simulation
MeSH	Medical Subject Headings
MIEM	Maritime Information Exchange Model
MIT	Massachusetts Institute of Technology
MMOG	Massive Multiplayer Online Game
MPL	Mozilla Public License
MSDL	Military Scenario Definition Language
NATO	North Atlantic Treaty Organization
NCR	Network Change Request
NCSA	National Center for Supercomputing Applications
NDR	Naming and Design Rules
NLM	National Library of Medicine
NPS	Naval Postgraduate School
NUWC	Naval Undersea Warfare Center
OEM	Original Equipment Manufacture
OOB	Order of Battle
OOP	Object Orientated Programming
OPORD	Operation Order
OPTASK	Operational Task
OSS	Open source Software
OWL	Web Ontology Language
PDF	Portable Document Format
PER	Packet Encoding Rules
PM	Program Manager
PMC	Project Management Committee

PMP	Portable Media Players
POR	Program of Record
POS	Point of Sale
PPL	Preferred Product List
PSVI	Post Schema Validation Infoset
QoS	Quality of Service
RDF	Resource Description Framework
RFC	Request For Comments
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RSS	Really Simple Syndication Format
	Residual Sum of Squares
RTI	Run-Time Infrastructure
SAX	Simple API for XML
SGML	Standardized General Markup Language
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPAWAR	Space and Naval Warfare Systems Command
SSAA	System Security Authorization Agreement
SSE	Systematic Software Encoding
SSIL	System/Subsystem Interface List
StAX	Streaming API for XML
STB	Set-Top Box
SVG	Scalable Vector Graphics
SWRL	Semantic Web Rule Language
T&E	Test and Evaluation
TAR	Tape Archive
TCE	Trusted Computing Environment
TDCS	Track Data Conversion Suite
TLP	Apache Top-Level-Project

TYCOM	Type Commanders
UAV	Unmanned Ari Vehicle
USD	Under Secretary of Defense
UTF	Unicode Transformation Format
VM	Virtual Machine
VMF	Variable Message Format
W3C	World Wide Web Consortium
WWW	World Wide Web
X3D	Extensible 3D Graphics
XBC	XML Binary Characterization
XBIS	XML Binary Infoset
XBRL	Extensible Business Reporting Language
XEBU	XML Efficient Binary
XEUS	XML document Encoding with Universal Sheet
XML	Extensible Markup Language
XMPP	Extensible Messaging Presence Protocol
XMS	XML Meta Structure
XSLT	Extensible Stylesheet Language for Transformations
XSBC	Extensible Schema-Binary Compression
XTC	XML Tactical Chat

ACKNOWLEDGMENTS

Thanks to my wife, Alice, for enduring NPS and me. You were cheated out of “shore duty” down time as I worked longer hours here at NPS than I did while attached to a deploying afloat command. Being “deployed at home” is a unique challenge, harder than physically being out of country. Without your strong support, our family would not have made it through this challenging time, and I would have failed academically, professionally and personally.

Parker, your “That’s papa’s school” comment every time we drove by NPS when all I wanted to do was hide from the vision of computer code kept me grounded on just how unimportant everything really is outside of our house.

Carson, “bye bye” is coming soon and “night night” will hopefully be in my future the day I leave Monterey. Unfortunately, “night night” will have to be in a hotel without all of you the first few nights.

Don Brutzman, I hope I was able to give you what you wanted. I guess if you sign this ever expanding thesis, some measure of success can be taken. Thanks for keeping your cool trying to explain the social networking EXI mail list thing to me. It took nine months of our weekly meetings, but TPAC got it done. If I take anything from my time with you, I hope it is your ability to manage what seems like an infinite number of simultaneous projects. How you do it, I cannot begin to guess, but exposure to your ways will help guide me in my future endeavors for the rest of my life.

Don McGregor, your simple and easy to follow code samples got me further than you may realize. You are an outstanding programmer and truly can teach computer programming. NPS is lucky to have you as an asset. Additionally, your comments on my thesis were always great and easy to follow. You clearly care and want to help us students succeed. Thank for your patients, experience, and time.

Professor Lynn Whitaker I owe you a ton for all of the insights you gave me into statistics and Data Mining. I had to go out of my way to work your class into my schedule, and I am glad I did. Thank you so much for dropping all you were doing to

help me check my inferences. I know you are busy and I truly appreciate the effort you gave me on a moment's notice. I only wish I could have worked with you more.

Taki Kamiya you have the patience and calmness I desire to achieve. Thank you for working with me and all of my basic question. I hope I was able to contribute some value to the process, and I hope the project comes to fruition, benefiting IT in general. You are the right person to lead the push, good luck.

Daniel Peintner thank you for answering my many EXI questions, such as what does the X in EXI stand for. Just kidding, but I do know some of my questions were far more basic than where you were currently working so I thank you for breaking your rhythm to help me. I hope the next time I see you that you have the prefix Dr.

EXI Working Group thank you for letting a novice like me join your meetings and listen in on your progress. It was a real eye opening experience to see how real-world IT does business. I will be following the progress and discussing EXI at my next job, the Navy's network operations center.

Also want to thank Web 3D Consortium as their membership within the W3C is what enabled me to meet with the EXI working group and join in the e-mail mailing list.

I. INTRODUCTION

A. PROBLEM STATEMENT

The ubiquitous use of XML as a data capture format has made it the unofficial standard format for data exchange in nearly all technology domains in both the business world and the Department of Defense (DoD). Specifically, the DoD has embraced XML as the mandated data representation format for the Network-Centric data sharing strategy in support of the Global Information Grid (GIG). This DoD XML strategy ensures all data is visible, usable, and most importantly, interoperable when and where needed to accelerate decision cycles. XML has achieved this level of adoption by being simple, platform independent and easy to use requiring relatively no formal training in order to get the spirit of XML documents using nothing more than a fundamental text editor due to XML's text-based and inherently self-documenting structure. However, these advantages of XML have unintentionally placed limits on the locations and devices that are able to deploy (send, receive or process) XML data. This DoD XML-based data approach comes at the cost of limiting real-time network edge device (small mobile devices) connectivity to the GIG because they are seldom able to meet the necessary bandwidth and processing requirements to process XML due to its intrinsic nature of being verbose and often complex to process. This thesis describes a powerful and robust solution to the XML limitations by means of the Efficient XML Interchange (EXI). Using EXI's compact and computationally efficient encoding of XML, the XML limitations on locations and devices are minimized and the expected ubiquitous future use of XML is preserved.

These XML's deployment problems will be exacerbated as time progresses due to the ubiquity of expected XML use within the DoD and technology sectors. If XML's known limitations are not overcome, due to the common place of XML use, its limitations present a potential for degradation on standardized data exchange to meet the future's needs. EXI is the World Wide Web Consortium (W3C) proposed standardized alternative XML format that presents a solution to overcome XML's limitations.

XML is a verbose text-based format by its original design and intent, which results in many benefits. This verbose text-based style enables humans to read and understand the data, even editing the data relationships within an XML document using nothing more than a simple text-editor. XML is platform independent because the data is text-based, removing the Endian problem common to binary formats which depend on which bit of each byte of information is the most significant between different processor architectures. However, this universal and functional data-representation style comes at a cost of large files and likely processor-intensive operations. XML documents are always larger than the actual data they represent because of the XML specification mark-up tags. Additionally, predominantly numeric XML documents become processor intensive due to frequent string-to-numeric conversions.

XML file size and processor complexities quickly become problematic for locations that do not have large bandwidth capacity or that do not have powerful processors. DoD relevant examples include deployed naval ships operating on an INMARSAT link (similar in speed to a legacy dial-up modem) for all off-ship network communications, or a Marine in theater using a handheld device as a language translator which consists of a small primitive processor along with limited battery capacity. Military relevant files are generally numeric, and when in XML format, often reach multi-megabit sizes with only a fraction of the file size being actual data. Remote mobile units commonly are unable to meet the bandwidth and or processor requirements to send and receive XML data.

The Efficient XML Interchange (EXI) encoding format presents a solution to both the excessive file size and processor intensive operations of XML documents. First, the EXI encoding exploits the structured nature of XML, recognizing many of the hierarchical tags, as well the information content repeats throughout the XML document. EXI replaces repeated content and tags with a compact alternative identifiers, which reduces the cost of document file size. Second, EXI represents numeric datatypes as compact binary values rather than text, which reduces the cost of string-to-numeric conversions during decompression and loading.

Through EXI, the Department of Defense (DoD) will be able to deploy XML-based data deeper into the battlefield, all the way to the individual mobile war fighters. This deeper network enhances war fighter's ability to know and share information of value, thanks to swifter and more efficient communications.

B. MOTIVATION

1. Extend XML to Unsupported Domains Cases

A primary goal of this thesis is to present a potential solution to extend the internet and other networked decision tools deeper into the battlefield, down to the individual mobile user and remote commands; to enable network communications at the network edge.

2. Mitigate XML's Weaknesses

Great benefits are possible and achievable from XML optimization, and XML is how the data is represented in nearly every digital domain. XML usage is prolific in both the DoD and the business world, and shows no sign of declining in popularity in the future. This thesis presents a method to mitigate XML's weaknesses in order to maximize the leverage of the diverse XML base.

3. Tactical Bandwidth and Small-device Improvements

Imagine you are in the middle of a battle and you have to make a decision that place life and equipment at risk. You insert the current situation into a handheld device that then relays the input parameters to Higher Headquarters (HHQ) where the parameters are loaded into a High Performance Computing (HPC) simulator. The simulator runs and analyzes 10,000 tactical replications, and then forwards the result of the analysis-of-alternatives back to your handheld device with a recommendation for the most likely successful Course-Of-Action (COA). In addition to the result of the simulation, the feedback to the handheld allows you to walk the tactical situation within a virtual 3D space, enabling you to gain an intimate familiarity with the recommended course of action before entering harm's way.

An example scenario might be a naval ship transiting a heavily fortified strait lined with Anti-Ship Missiles (ASM). The simulation could provide the most optimal speeds, courses, course changes and time of day to navigate the strait safely.

An additional scenario might be the rescue of hostages with a squad of soldiers. Running the scenario based on the layout of the hostage location, the number of soldiers and the expected weapons of both the friendly and hostile forces, the best deployment of all soldiers can be predetermined, as well the most likely location of the hostages. Again, through a 3D representation of the simulation result, the soldiers can visually rehearse their mission before conducting the hostage rescue.

These scenarios sound like science fiction, but are feasible today if research capabilities are deployed. The limiting factors are the handheld devices due to the size and complexity of the simulation result file and low-bandwidth connectivity in the battlefield. Handheld devices are capable, but bounded in their potential by heat dissipation ability, memory capacity, battery life, as well the interaction of these bounds. All of these limits can be overcome by using the EXI compression for XML-based data as this thesis discusses.

C. 2-MINUTE ELEVATOR SPEECH ABOUT THESIS

An end state of this thesis is to show how to integrate EXI into DoD networks in order to push network traffic and the Network-Centric data sharing vision deeper into the low-bandwidth battlefield networks, specifically to limited memory, low power, and micro-CPU handheld devices. Figure 1 shows several interagency high-bandwidth capable land stations communicating a distributed simulation in real-time with a low-bandwidth mobile soldier and deployed navy ship; overcoming bandwidth and micro-CPU limitations through the integration of EXI for XML-based communications.

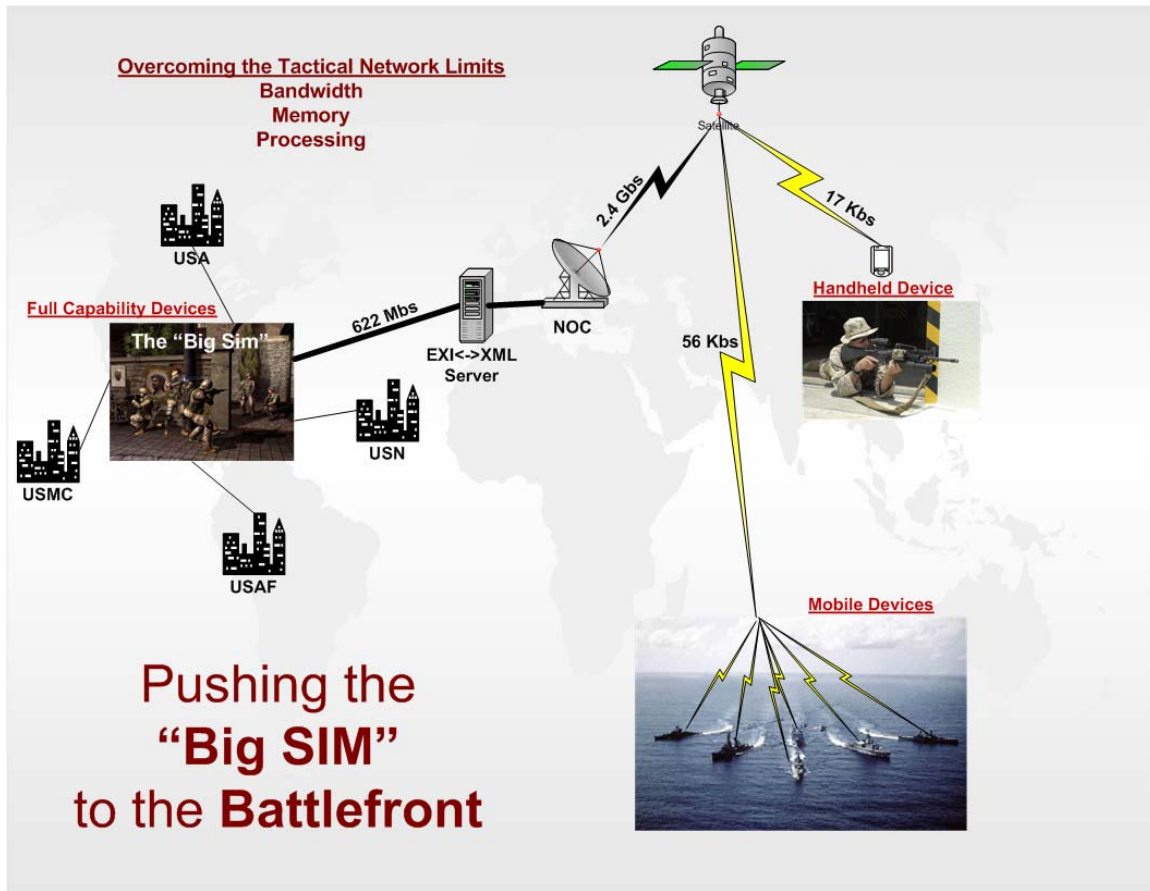


Figure 1. EXI Enabling Real-Time High and Low-Bandwidth Locations to Interoperate

D. RESEARCH QUESTIONS

Several research questions are addressed within this thesis:

1. **Can the Department of Defense (DoD) Keep Up with the Business World's Constantly Connected Internet Applications Philosophy?**
 - The business world is operating under the assumption of a constant and high-speed internet connection. For example, some companies are farming repository space so their clients do not have to store any data on their local hard-drives, just connect via the internet and the data is present.
 - Ships and mobile forces cannot maintain full-time connections to internet.

- Bandwidth is minimal in the battlefield or at sea, and platforms cannot support high data-rate requirements.
 - Handheld devices cannot process computationally expensive documents due to limited battery and inherently limited processor capabilities.
- 2. Can DoD Data, using the Extensible Markup Language (XML), be Efficiently Compressed to a Level that Makes Porting to and from Low-bandwidth Military Units Feasible?**
- XML is an industry standard format for data representations and exchange.
 - The DoD mandates XML as the data exchange format for future system developments.
 - Managing XML is going to become an even larger bandwidth issue in the future.
- 3. What are the Risks to the DoD Infrastructure if Methods to Push Data Further Down the Echelon Chain of Command are Not Developed?**
- Network-Centric Warfare is centered on network communications from many sources simultaneously; every sailor and soldier a sensor.
 - Timeliness of information is the key to modern warfare; future wars demand wide dissemination of information to large numbers of simultaneous mobile users in real-time.
- 4. Is XML an Effective Tool for Commands that are Under Extremely Low-Bandwidth Constraints?**
- What benefits aside from data interoperability does XML bring to those commands operating under extremely low-bandwidth that warrants XML consideration?

- Can XML-structured formats be leveraged to extend message exchange and interoperability with units operating under extremely low-bandwidth conditions?

E. SCOPE OF THESIS

The primary scholastic focus of this thesis is the study and analysis of a solution for eliminating XML limitations in DoD networks by the use of EXI. This study is conducted with statistical experiments of EXI's impact on tactical DoD messaging improvements. Several statistical predictive models are developed to aid future EXI implementers and measurement tools to evaluate the potential effect of EXI within their domain.

This thesis explores the history of XML: where XML is found, why XML is important to DoD, where XML is going in DoD's future, and why XML might become a burden to DoD if the XML file size and processing complexities are not properly considered in future data and system acquisition decisions.

Next a discussion and recommendations for development and integration of an EXI solution is presented, focused on an open source implementation for Web-centric integration. Also addressed are the expected administrative challenges that an EXI solution must overcome before being authorized for installation within DoD and other networks.

A Java-based implementation of the EXI specification, called OPENER-EXI, is started in support of this thesis. "Opener" in the title is to indicate an open source implementation of EXI, that is not viral, free to use and distribute as seen fit by the end-user. The scoped focus for this implementation subset is a schemaless byte-aligned EXI encoding.

F. METHODOLOGY

1. Background Study

This thesis starts with a study of the XML design principles in order to gain an understanding of reasons for the XML format. This initial study helps explain why XML is a powerful data representation format and highlights XML limitations of verbosity and processing complexities.

2. Alternative XML Formats

The thesis continues with a review of the technology sectors attempts to overcome XML's limitations and the W3C's efforts to establish a standardized alternative XML format, Efficient XML Interchange (EXI).

3. Efficient XML Interchange (EXI) Deployment

A DoD focused EXI deployment strategy is presented, addressing licensing, DoD system integration and DoD network security concerns. Recommendations for an Open-source License scheme tailored towards a Web server integration are presented to enable the widest and fastest employment of EXI into the DoD networks.

4. Efficient XML Interchange (EXI) Development

Software implementation considerations and design principles are presented along with the Naval Postgraduate School's initial EXI implementation. This initial development is Java based and addresses the EXI algorithms.

5. Statistical Comparison and Modeling of EXI Performance

A study of EXI's performance within DoD is conducted, showing that EXI delivers both better compression and performance compared to existing DoD network data sharing methods. Presented are statistical predictions models as tools for EXI implementers to evaluate EXI's performance before integration.

6. Conclusion

This thesis concludes that the DoD has a doubling of bandwidth potential for XML-based data exchanges. Since the Web is the center of many networks and often are founded on XML, a Web focused integration of EXI has the maximum benefit for DoD. Ultimately, through EXI, the DoD will be able to meet its Network-Centric data sharing strategy at the network edge, delivering a more timely informed force.

G. THESIS ORGANIZATION

Chapter II – BACKGROUND AND RELATED WORK. This chapter covers the governing body of XML and the basics of compression, encryption, and packaging. Several key terms used within this thesis and several text-based compression tools are defined.

Chapter III – XML RELEVANCE. This chapter provides a brief overview of the XML's basic design. It is followed with a summary of the importance XML plays in the Information Technology (IT) world today. The DoD mandate for XML usage is discussed, as well as several cases study of Tactical XML usage highlighted.

Chapter IV – SHORTFALLS OF XML LEADING TO EXI. This chapter points out that while XML is a powerful and proven tool, its file size and verbosity is a weakness that prevents XML's tactical use at the mobile user level, and in low-bandwidth environments. Both are environments where DoD predominantly operates.

Chapter V – BINARY XML FORMAT RATIONALE: XBC. Given XML's shortfalls, this chapter describes the developmental background of the efforts to produce an efficient and compact XML format by the XML Binary Characterization (XBC) working group of the World Wide Web Consortium (W3C). Several years of combined efforts from individuals and corporations around the world contributed to that work, helping define precise metrics and capabilities for an efficient XML format.

Chapter VI – W3C BINARY XML FORMAT (EXI) DECISION

JUSTIFICATION & FRAMEWORK. This chapter discusses the methodology used by the W3C Efficient XML working group in their selection of EXI as the alternative XML format, highlighting the evaluation process used by the W3C. This chapter also discusses the impact and recommended usage of EXI.

Chapter VII – OPENER-EXI IMPLEMENTATION RATIONALE. This chapter lays out a roadmap of recommendations for integrating EXI into DoD networks by addressing licensing, system security requirements, and integration planning. The administrative constraints and policies to implementing an EXI solution into DoD are further discussed.

Chapter VIII – EXI STRUCTURE & OPENER-EXI IMPLEMENTATION. This chapter discusses the EXI technique encoding and decoding XML documents through sample code examples and simple test-case documents. This chapter also describes the OPENER-EXI development rationale.

Chapter IX – DEMONSTRATION & ANALYSIS OF RESULTS. This chapter describes the test-corpus used to evaluate the effectiveness of the EXI on DoD XML needs. Results of EXI applied to DoD relevant text cases are presented and summarized with statistical models and analysis of results. A discussion of available EXI implementations and tools is also discussed.

Chapter X – CONCLUSIONS AND RECOMMENDATIONS. This chapter summarizes the overall effect of the analysis of results and expected impact to DoD. This chapter also provide recommendations for future work.

II. BACKGROUND AND RELATED WORK

A. INTRODUCTION

This chapter provides an overview of diverse topics that provide information relevant to XML compression. Topics covered include the controlling organization of the XML specifications, XML's design points and the definition of key terms and concepts. Also discussed are several common file compression techniques.

B. WORLD WIDE WEB CONSORTIUM (W3C)

The World Wide Web Consortium (W3C) develops a variety of interoperable technologies, specifications, guidelines, software, and tools to lead the Web to its full potential (W3C, n.d.). W3C is a forum for information, commerce, communication, and collective understanding.

C. XML IN 10 POINTS

The basic concepts of what XML is and its design goals are contained within the W3C's (1999) *XML in 10 Points* document:

1. XML is for Structuring Data

XML organizes the data it describes hierarchically through parent-child relations using a small set of rules. XML is not a programming language, it defines data; XML makes data reading and writing by computers simpler without ambiguity by means of the platform-independent Unicode.

2. XML Looks a Bit Like HTML

Both HTML and XML use tags to delimitate data boundaries (words bracketed by '<' and '>') as well as attributes (of the form name="value"), but unlike HTML, XML does not define what each tag means or how the data within the tag should be represented. XML leaves the interpretation and representation of data to the

implementing application. Additionally, XML is not forgiving to structural errors such as missing tags or attributes without quotes. HTML can make assumptions about the intent of a document when formatting errors occur. XML, however, will not make assumptions about data intent when formatting errors occur, throwing an error by definition of the XML specification.

3. XML is Text, but is Not Meant to be Read

Unlike other data storage techniques that use a binary format for data representation, XML uses a text format, which enables users with nothing more than a rudimentary text-editor, and not necessarily the same editor that created the XML, to view and modify XML. Generally, humans should not directly read XML, though the capability is present.

4. XML is Verbose by Design

Due to XML's text format and tagging method, XML documents are usually larger than comparable binary formats. However, the XML designers did this consciously.

5. XML is a Family of Technologies

The XML 1.0 specification defines what "tags" and "attributes" are, not what the data contained within them means. This enables XML to be extended for any purpose allowing developers to define the meaning of data as needed; <p> denotes a paragraph in XHTML, but might be a person or place in another XML language.

6. XML is New, but Not That New

Before XML there was the Standardized General Markup Language (SGML) developed in the early 1980s, an ISO standard used for large documentation projects. Then came HTML started in 1990 as a child of SGML. SGML is a markup language, but to novice users is difficult to understand.

XML designers took the best design aspects of SGML and HTML and produced XML. Design goals were to make XML no less powerful than SGML, but more regular and simpler to use. SGML was narrow in focus, but XML is more general and abstract in its applicability. XML officially started development in 1996 and became a W3C recommendation February 1998. It has remained stable with only minor refinements for internationalization (I18N), etc., since then.

7. XML Leads HTML to XHTML

XHTML, the successor to HTML is an XML technology. XHTML has many of HTML's elements, but the syntax has been updated to conform to XML rules, such as XML's strict structured compliance requirement. XHTML restricts XML by defining certain tags that can only be used for specification-defined purposes. The <p> tag in XHTML means paragraph and cannot represent anything other than a paragraph.

8. XML is Modular

XML allows for document format combining and reuse. To prevent the overloading of identical tags, where "<p>" means "paragraph" in XHTML while indicating "person" in another XML language, XML provides a namespace mechanism. Each unique format defines its own namespace to prevent overloading of common tag names.

9. XML is the Basis for RDF and the Semantic Web

Resource Description Framework (RDF) is an XML format for describing data ontologies for the Semantic Web. RDF enables data merging between heterogeneous sources regardless of underlying schema, and supports data format evolutions without requiring changes to be made replicated across all data sources (W3C, 2010). XML is Web orientated and so is naturally aligned with RDF.

10. XML is License-Free, Platform-independent and Well Supported

Directly quoted from the “XML in 10 Points” (W3C, 1999):

By choosing XML as the basis for a project, you gain access to a large and growing community of tools (one of which may already do what you need!) and engineers experienced in the technology. Opting for XML is a bit like choosing SQL for databases: you still have to build your own database and your own programs and procedures that manipulate it, but there are many tools available and many people who can help you. And since XML is license-free, you can build your own software around it without paying anybody anything. The large and growing support means that you are also not tied to a single vendor. XML isn't always the best solution, but it is always worth considering.

These points all remain true and become even more relevant with the expansion of XML impact via EXI.

D. DEFINITIONS

1. Focal XML Points within this Thesis

This thesis is concerned with all ten points of XML, but primarily focuses on two of the points to explore compression and efficiency improvements for the XML family of extended languages: Point 3 that “XML is text, but is not meant to be read”; and Point 4 “XML is verbose by design.”

2. Handheld, Mobile and Micro Devices

Specific network devices of interest for this thesis, beyond desktop computers and servers, are handheld, mobile and micro devices. While these terms each define specific subsets of small network capable devices, they are used interchangeably throughout this thesis based on the common assumption that they all have small CPU capacity, limited memory capacity and are bandwidth limited. Examples of such devices are cell phones, appliances, or a mobile survey device similar to what a gas and electric technician might carry to take utility readings at a residence or business. While effort is maintained to use the term “handheld devices” for consistency throughout this thesis, occasionally the terms

“mobile” and “micro” are used when applicable, but with the same intent.

Fundamentally, specific hardware differences are not of concern within this thesis, just the concept of enabling limited capacity hardware to do more using compression and efficiency techniques.

3. Compression

Compression is the application of techniques to a file that reduces the size of the file. Information compression comes in two broad categories of lossy and lossless, where the former does not decompress back to the identical original file and latter does. The choice of lossy or lossless is specific to the domain application. In many cases, a lossy compression is perfectly acceptable, such as JPG imagery where image quality remains acceptable, and in others cases lossy it is not acceptable.

For this thesis, the concept of compression enables more data transferring under a predefined bandwidth condition. Compression also enables greater file storage capacity when archiving XML information.

4. Encryption and Digital Signature

Encryption and digital signature are different yet related to compression, and briefly highlighted here for thoroughness. While compression uses the fewest bits possible to represent a file, encryption and signature tend to add bits to the file, by means of combinatorial techniques in order to increase the entropy of the file, making the raw data unreadable without a token key to decipher the data. For example, ASCII character encoding uses 8 bits delivering 256 possible characters. A basic encryption algorithm might project ASCII 8-bit to a higher order 20-bit encoding, delivering 1,048,576 possible characters, but only the original 256 from the 8-bit are of value. The 12 additional bits per character are pseudo-randomized to add complexity, increase the entropy, and in doing so, makes the file unreadable to unauthorized recipients.

5. Packaging and Archiving

Packaging, also called archiving, is the collection of files and merging them into a single collection file. This is common for digital media distribution such as in the form of .iso files for CD images. Compression can be applied to a package as a whole, to the individual files within the package during packaging, or as in the .iso file format for disks, with no compression. Packaging enables the transfer of multiple files as a single transaction.

6. Text-based Compression Generalized

The fundamental aspect of text-based compression is the replacing of redundant text data with a smaller representation.

In the simplest of formats, text-based compression techniques catalog all unique text entries (tokens) that are contained within an input text file. The output compressed file is simply a list of indexes into the catalog. Each index in the output-compressed file represents a piece of text in the original input file, but as a pointer into the catalog. The indexes are smaller than the original text, which is what delivers compactness overall, so long as enough tokens repeat to reduce total size used.

Normally, a window of opportunity is defined for a compression technique that limits the range of the input file from which the catalog can address its entries. The window is reapplied across the file, rebuilding the catalog at each step. At each step of the window, until the end of the input file is reached, the catalog along with the indexes of the underlying input file is written to the compressed output file.

A window instead of the entire document is used in order to capitalize on the likelihood relatively close text blocks should have more commonality than text blocks that are further apart. A window sets the range to find redundant tokens while keeping the total number of tokens small. The fewer the tokens, the more compact the information can be formatted. This approach prevents catalogs from being excessively large, but at the possible expense of lost redundancy among subsequent catalogs.

A common index size metric, often in terms of bits, is the $size = \log_2 n$ equation, where n is the number of entries in the catalog. If there are 32 entries in a catalog, $n = 32$, then only $5 = \log_2 32$ bits are needed to represent all possible indexes. Given that every text character is 8 bits and every text token is likely to have multiple characters, at least a 3-bit savings per token is achievable if every token in the worst case consisted of a single character. The larger the window size, the more likely the number of entries in the catalog will increase, which in turn increases the index size and ultimately reduces compression efficiency.

Generally, what distinguishes one compression technique from another is the way they manage the window of opportunity and how they computing the indexes that refer back to the catalog.

E. RELATED WORK

1. GNU Zip (GZip)

GZip is an optimized, lossless, and open source compression utility created to be a general replacement of existing compression techniques (GZip, 2003). It has been widely adopted on servers, such as HTTP, to optimize traffic flow as shown in Figure 2 (IETF, 1999). GZip optimization is achieved by requiring only a single pass through the file without the need for backwards seeking, and does so without knowledge about the input media type, or file size. The result of GZip is a file renamed with the .gz extension.

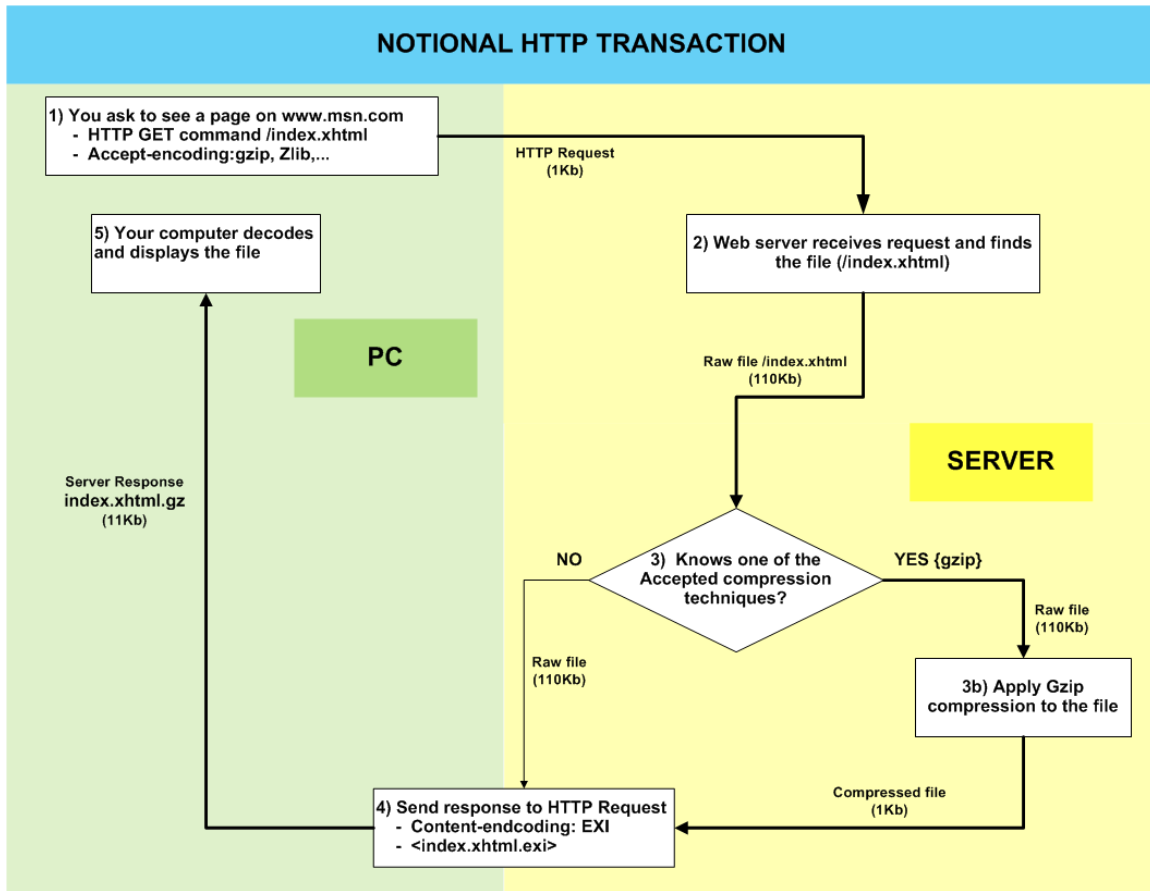


Figure 2. Notional Step-by-Step Example of Network Throughput Optimizing by Negotiated Compression

GZip is commonly used within DoD at the server and desktop level, and so is subject of further study within this thesis. Additional information about GZip can be obtained from the GZip source page at www.gzip.org, and specifics about the file format of GZip is described in the Request For Comments (RFC) 1951 and 1952. Usage of the GZip algorithm is royalty free.

2. Zip

Zip or PKZip, developed by Phillip Katz in 1984, is the ubiquitous compression technique on the market (PKware, n.d.). It operates on the same basic algorithm principles as GZip, but unlike GZip, the Zip routine allows for the packaging or sometimes called archiving of multiple files into a single compressed file (PKware,

2007). Zip is able to compress a single file, multiple files, a directory, or combinations of files and directories while retaining the original input file-structure.

The Java Archive (.jar) file format uses Zip to package its collection of classes, thus .jar files can be read with a Zip tool (Sun, n.d.).

The general compression encoding flow of Zip, as well as any other packaging technique is to sequentially encode the input files starting at index point 0 relative to the Zip output file. After all files have been compressed, a central directory is written to the end of the Zip output file listing the encoded files along with their relative starting address. Zip file referencing is demonstrated in Figure 3, borrowed from Wikipedia (Zip, 2010).

At decode time for a particular file, its relative index is retrieved from the Zip's central file directory. A file pointer is moved to that index and the compressed file read applying the Zip decompression technique until the End-of-File (EOF) mark is reached. The results is a new decompressed output file that is identical to the original input file.

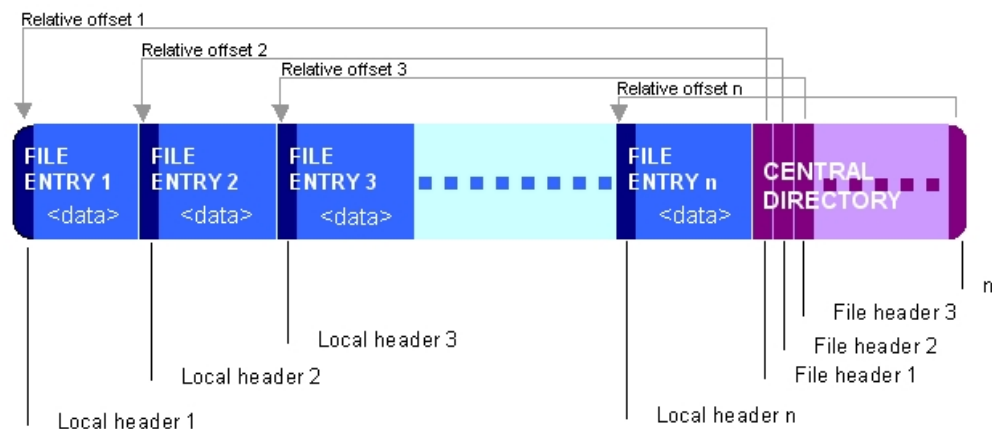


Figure 3. Zip Archive Notional File Structuring (From Zip, 2010)

There are a number of methods to conduct Zip compression with various limits on each. Key points to check are the file size limits (32 bit = 4Gb or 64 bit = very large)

imposed on the encoding based on indexing size. Zip is the most common desktop compression tool available in DoD, and as such, is subject of further study within this thesis.

3. 7-Zip

7-Zip is another compression utility that uses the best of archiving and compression in one user-friendly utility, Figure 4, and it does so with superior results compared to both GZip and Zip. The 7-Zip technique uses the Lempel-Ziv-Markov Chain-Algorithm (LZMA), which is an improved and optimized version of LZ77, essentially optimized GZip (7-Zip, 2009). The improvements over GZip come from adaptive indexing on the length and distance tokens to identify each unique byte within the input file.

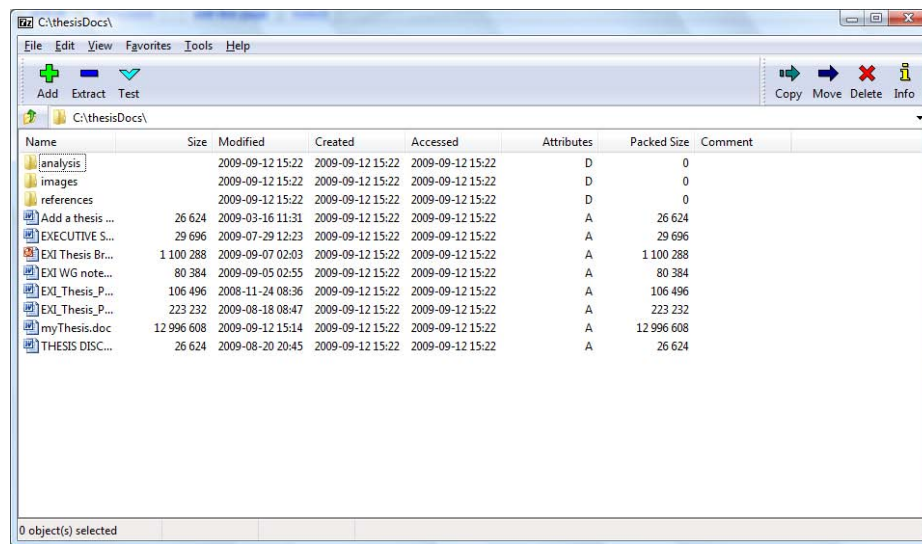


Figure 4. Example 7-Zip File Manager User Interface

The 7-Zip technique is not available within DoD and not authorized for use so it is not studied directly within this thesis. However, for a highly capable, versatile and configurable compression tool, as shown in Figure 5, 7-Zip is highly recommended. 7-Zip is licensed for free distribution under the GNU LGPL so its royalty free for any use. Go to <http://www.7-zip.org/> for additional information about the 7-Zip technique or to download the source-code (Java, C++, C# and C) and binaries.

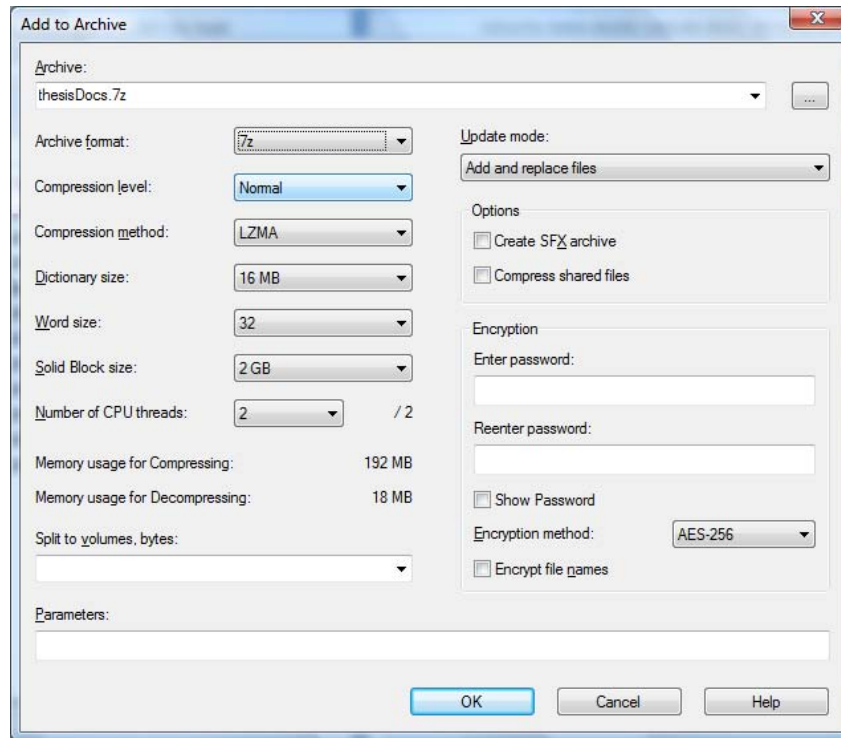


Figure 5. Example 7-Zip Output Configuration Settings Interface

4. Efficient XML Interchange (EXI)

The Efficient XML Interchange technique is an XML-specific compression format that is the subject of direct study within this thesis. Chapter VIII of this thesis covers the EXI specification in detail and Chapter IX demonstrates the effectiveness of EXI techniques.

5. XMill

XMill is an open source XML-specific compression technique that can generate compression levels better than GZip, freely downloadable from <http://sourceforge.net/projects/xmill/> (Liefke & Suciu, 2000). The superior compression is achieved through a container grouping strategy that collects and compresses text items together based on their semantics, parent label and tag path relationships. This generally leads to higher compression ratios since each group is more likely to contain text items with many similarities relative to document wide relations. Additionally, a user can also

specify how to “pre compress” a specific text item. For example, the user might want to replace the 'Age' string with its binary integer representation or an IPv4 number could be replaced by four bytes.

After the input XML document is containerized, a conventional compressor such as GZip is applied to the containers. Since the number and size of containers grows with the size of the XML document, a memory-window mechanism is implemented. After the overall size of the containers reaches a certain user-modifiable memory window size, the containers are compressed and stored in the output file.

The results of XMill are stored with the extension .xmi for .xml files and .xm for all other files in the XML family of languages.

Even though XMill generates superior compression levels compared to GZip, and in many cases even better than EXI, XMill is not a focus of this thesis due to several technical issues with the code setup during initial testing and evaluation. Further, XMill is not an authorized DoD application. Specific technical problems with the current implementation:

- Inability to work on list types. When attribute values are of the List type, array of values, the XMill application views the extra white space as errors, expecting a unique attribute declaration for each value of the list. An example of a list datatype is the position attribute of an X3D Viewpoint node, a node that sets the position of a camera in 3D space for the X3D language.

```
<Viewpoint position='-61.07 8.139 -33.28' />
```

- Improperly handles forward slash “/” characters within value fields. When the forward slash character is used within an attribute value field, it causes tokenization errors in XMill, indicating XMill is expecting an end-element declaration. An example of this is an attribute’s value content that is an uri using the ‘/’ character as the file path separator.

```
<x:xmpmeta xmlns:x='adobe:ns:meta/'>
```

6. Tar

Tar is one of the original compression and packaging utilities created for the UNIX environment when secondary computer memory was on tape-drives, which is where TAR gets its namesake: Tape Archive (Tar, 2009). The Tar technique enables archiving with or without compression of one file/directory or many files/directories, and does so while maintaining the original file-structure.

Tar can still be found in uses within the UNIX and server environments, but today this use is essentially a GZip technique combined with the ability to package multiple files/directories. Compressed Tar files are typically given the extension .tar.gz or .tgz. Although it is in widespread use, Tar is not an authorized DoD application and so is not subject of further study within this thesis.

F. CHAPTER SUMMARY

This chapter describes a wide range of topics that are relevant to XML-based compression. The governing organization of XML is introduced as is as the design points of XML's creation. Specific examples of compression and related aspects are introduced as well are some of the key terms used throughout this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

III. XML RELEVANCE

A. INTRODUCTION

This chapter describes the basis of XML in terms of what it is and how it accomplishes what it does through its structured format. The chapter continues by exploring the depth of XML penetration within both the business world and DoD, and concludes with several case studies of tactical XML usage within DoD.

B. GENERAL XML FORMAT OVERVIEW

The Extensible Markup Language (XML) is a tagging language used to describe the structure as well as the data contained within a document (Carey, 2007). The word “extensible” means XML is capable of modifiable design to match any arbitrary data set. Such data sets can include anything from personal contact list or spreadsheets, to technical drawings within a complex CAD specification. The general principle governing the XML family of languages is to define meaningful tagging of the data with human readable text (Hunter, Cagle, Dix, Kovack, Pinnock & Rafter, 2001). Thus, XML itself is a Meta language, i.e., a language for describing other languages. XML’s design is focused on the data, not the processing of the data.

Tags surround the data they are describing with meaningful (self-documenting) tag names enclosed within angle brackets:

- The start of each data node
 - <Self_Documenting_Meaningful_Name>
- The termination or scope of each data element
 - An exact case sensitive match of the start tag, but with a forward slash at the beginning of the name declaration
 - </Self_Documenting_Meaningful_Name >

The XML code example in Table 1 from the W3C *EXI Primer* document depicts EXI's "Hello World" code example, the notebook.xml document (W3C, 2007).

```
<?xml version="1.0" encoding="UTF-8"?>
<notebook date="2007-09-12">
  <note category="EXI" date="2007-07-23">
    <subject>EXI</subject>
    <body>Do not forget it!</body>
  </note>
  <note date="2007-09-12">
    <subject>shopping list</subject>
    <body>milk, honey</body>
  </note>
</notebook>
```

Table 1. Simple XML Document, EXI's Unofficial "Hello World" (From W3C, 2007)

C. BREADTH AND DEPTH OF XML

Unlike XML's predecessor SGML, even without XML experience, anyone with a computer and a text-editor can open an XML file and interpret the data contained within the document. Given the example notebook.xml file above, the intent and content of the file is reasonably easy to obtain. It contains two notes called "note," each taken on different days in September of 2007, and each with a subject and body (message). Because of this easy-to-read and understand data format, nearly every avenue of the Information Technology (IT) industry has embraced XML. The architect of HTML, Tim Berners-Lee (1999) notes that XML was developed with nearly all the functionality of SGML, but with almost none of the complexity. Because of XML's simplicity, stability and robustness, the XML format has been adopted into configuration files, data set definition schemas, and any other data-representation function that required data persistence or parsing.

1. Business World XML-Specifics

Competitors in the business world often earn their money by being faster, more accurate, and most importantly, more efficient and adaptable than their competitors. XML is routinely the data medium that empowers the faster and better effects that business world corporations pursue.

Commercial applications require some form of initialization at run-time. This initialization can be hard-coded into the application, but hard coding makes an application difficult to change, requiring complete source code recompilation with each change. Therefore, most developers use an external configuration file instead of hard-coded initialization data that is modifiable external of the application, enabling changes without requiring the recompilation of the source-code. An example of this approach might be a word processor's tool bar preferences that remain persistent between uses. The existing preference information state at application shutdown is written to the processor's configuration file. This ensures persistence between application runs and the next time the processor is started the toolbar settings are the same as the previous run. Such configuration files have embraced XML for XML's descriptive and self-documenting format over the legacy .ini and .config non-interoperable proprietary formats (Carey, 2007).

The leading Computer Vision Artificial Intelligence packages, OpenCV, built by Intel, enables programmers the ability to do real-time video processing such as face detection for homeland security tools and unmanned aerial vehicle (UAV) target detection (Bradski, 2008). OpenCV uses the XML format to store its persistent file-structures as well as all of its configuration files for advance face-detection techniques. Using XML as the configuration file format, OpenCV software engineers are similarly able to make modifications quickly to facial-detection techniques without external support software. For example, facial-detection parameters can quickly be altered to suit the environmental conditions without recompiling the source-code, making OpenCV dynamically adaptable to environmental conditions, special facial features or clustered environments on demand as the tactical situation dictates.

Arguably, the most well-known and widely used office automation suite in use around the world is Microsoft's Office suite. Beginning with Office 2000, and some portions prior, Microsoft has used XML as the structure behind every application in the suite (MSDN, 2008). For example, every Excel document created is essentially a set of tagged text behind the scenes with a cell as the basic repeating structure (Wakenbach, 2003). This same general philosophy is found within Access (Prague, Irwin & Reardon, 2003) and in other forms for all the other Office applications.

The competition to Microsoft in terms of office automation is Sun's Open Office, which has been using XML as its backbone structure from its initial design consideration (Brauer, n.d.). Internal compression within Open Office data formats uses the Zip algorithm. Additionally, the Open Office formats are royalty free.

Within the business world, XML is found and utilized across the gamut of applications. From rudimentary configuration files, data archive, complex engineering data-structures, to the real-time application backbone format.

2. DoD XML Mandates Specifics

Like the business world, DoD has embraced XML, and has adopted it in a number of capacities. In support of the Network-Centric data strategy, XML is the backbone of the Global Information Grid (GIG) to deliver the future's expected needs and heterogeneous system-to-system interoperability (DoD CIO, 2003). DoD is focused on the value of data and the power XML structured data delivers an enterprise IT solution.

The Center for Strategic and International Studies in their recommendation to the 44th President in regards to Cyber Security, specifically arguing that standards and guidance for securing the internet be created, "The president should take steps to increase the use of secure Internet protocols" (Langevin, McCaul, Charney & Radvege, 2008). XML is the mechanism to accomplish this recommendation as XML is the ubiquitous format for communications for the World Wide Web (WWW) and beyond such as cellular phones (KDDI R&D Labs, 2003) and tactical military (Mohan, 2002) communication domains.

DoD-wide XML directives have come from the Secretary of Defense in a number of published documents mandating the use of XML. In the early stages of XML adoption to meet the Network-Centric data strategy, the DoD CIO (2004) published a checklist for Program Managers (PMs) to use to help steer them towards understanding data to empower them with the tools to make well-informed program decisions. Specifically, PMs are provided guidance on how to drive towards the Network-Centric vision of a Service-Orientated Architecture (SOA) GIG within their programs of responsibility using “tagged” data formats, that is, XML.

The Washington Headquarters Service, a provider of operational and administrative support services for the Secretary of Defense, lists XML as the preferred format for data transfer of a text-based nature (WHS, 2008 & 2010). This document does list several alternative formats, but places a number of conditions that must be met before an alternative to XML may be selected. XML, however, is the unconditionally prescribed data format.

The Assistant Secretary of Defense (ASD) has provided specific recommendation with regards to the use of XML, stating that the “Extensible Markup Language (XML)-based discovery metadata is the most flexible means of sharing discovery metadata throughout the Department of Defense” (ASD, 2006). This document goes on to provide steering guidance to all Program Managers (PM) on XML incorporation. The following year, the ASD provided requirements that are more stringent stating “...DoD PMs shall use Extensible Markup Language (XML)...” (ASD, 2007).

The Under Secretary of Defense for Acquisition, Technologies and Logistics (ATL) eliminated the use of the Military Standards System (MILS) and stated “Effective January 1, 2005, all information exchanges among DoD systems shall use the DLMS ANSI ASC X12 or equivalent XML schema for all business processes supported by the DoD 4000.25 series of manuals” (USD, 2003). The ATL alignment with an XML format enables the acquisition community to couple closer with the business world’s way of doing business, XML.

Various sub components of the DoD have also taken notice of XML. For example, the person within the Department of the Navy (DON) responsible for digital information is the Chief Information Officer (CIO), who has used this XML guidance to develop a number of XML policies and guidelines for the DON. The Chief of Naval Operations (CNO) understands the value of XML and supports the DON CIO initiatives quoted referencing XML as the means to information superiority on the sea (DON CIO, 2008):

The Department of the Navy will fully exploit [the] Extensible Markup Language as an enabling technology to achieve interoperability in support of maritime information superiority. CNO 2002

A presentation titled “DON XML - Achieving Enterprise Interoperability” summarizes the actions taken by the DON CIO to meet these goals and expectations (Jacobs, 2008). The first formal XML policy was written in 2002 creating an interoperability working group within the DON (DON CIO, 2002), and has been revised over the years expanding its vision and scope (DON CIO, 2005). The DON CIO has made it clear that the DON will use XML stating, “DON must be proactive about providing inputs to the standards bodies and solutions providers” which resulted in the DON Namespace document (DON CIO, 2005). The general mantra for DON CIO is to use XML whenever it is practical (Jacobs, 2008).

Specifically focusing on the secure internet and Network-Centric recommendation to the 44th president, the DON, by order of the DON CIO, plans to be aligning with a Service Oriented Architecture (SOA) by the year 2016 (DON CIO, 2008). Essentially, this equates to a XML-based Department of the Navy.

The DoD in general, much like the business world, has acknowledged the functional use and power of XML by mandating the transitions in full to XML-based technologies. Every agency within the Department of Defense has taken actions similar to that of the DON, which is essentially to transition to XML by a particular date in the near future. Some have gone as far as to having already transitioned in full such as the Under Secretary of Defense for Acquisition, Technologies and Logistics (ATL). The overwhelming argument for XML in DoD is interoperability and a consistent Joint, both

inter service and coalition force, data format. XML is the DoD vetted and chosen vehicle to ensure the Network-Centric Warfare vision is achievable to enable DoD to maintain its information superiority and dominance on the battlefield.

D. TACTICAL XML RELATED WORK

XML has found its way into numerous tactical environments throughout the DoD in support of existing systems, and the development of the next generation of systems.

1. Data Interoperability via XSLT Conversions

Early XML research focused on the use of XML as a middleware approach to enable heterogeneous systems to communicate between one another. Prior to XML, before any set of heterogeneous systems could communicate and share information collaboratively, both had to be able to translate between the sending and receiving systems' data format structure. For any two systems, translations would be a simple exercise of programming the translations directly into both systems. However, given the enormous number of systems in use within the DoD and other Government Organizations (GO), internal translation between every possible set of systems is impossible to directly code.

Using XML's Extensible Stylesheet Language Transformations (XSLT), a source application can share its data with a remote application without knowing the exact format of the other's data-structures, datatypes or any internal design. This is due to XSLT's ability to transform the contents of any XML document to another format (Carey, 2007) (Hunter et al., 2001). The general paradigm behind this concept is simple:

1. A sending system writes to an XML file the data it intends to share with a receiving system using its own data definition methodology.
2. An intermediate station translates the sending data using XSLT to a format, XML or other format, readable by the receiving system.
3. The transformed format is loaded into the receiving system and processed.

The data sharing process is accomplished without either side directly knowing any details of the other by means of the XML format and XSLT (Abbassi, 2003). The fundamental XSLT process is depicted in Figure 6. It is important to note that XSLT is XML, a member of the family of XML languages.

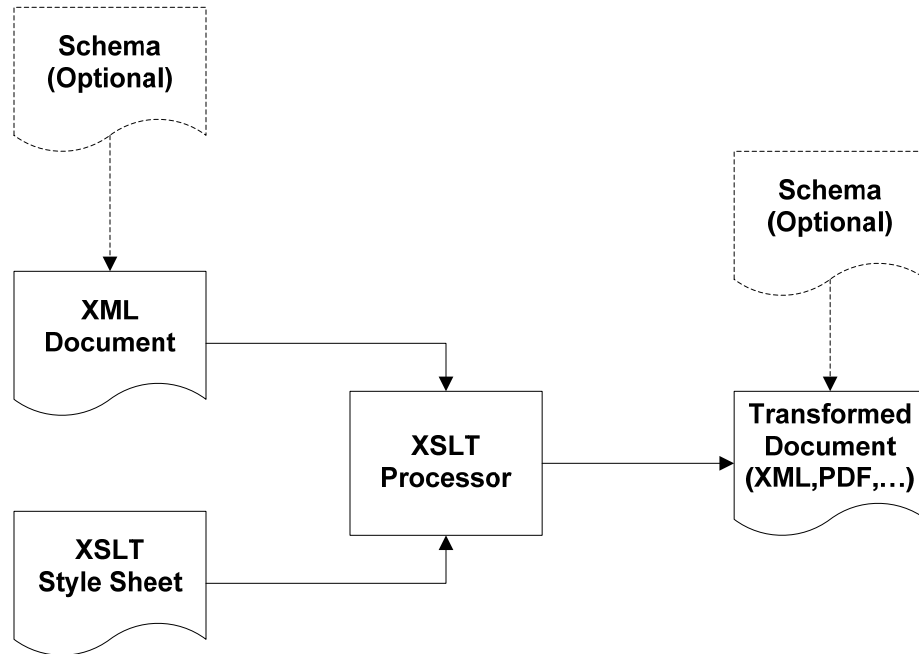


Figure 6. Notional XML's Extensible Stylesheet Language Transformations (XSLT) Process flow

Motivation for heterogeneous data interoperability is to enable legacy stovepipe systems to collaborate with modern systems. Pradeep (2002) explored this concept for legacy database systems. Using the basic recipe of the paragraph above, he found that transformations between arbitrary database architectures is possible using XML as an intermediary. Even if two databases are of the same relative architecture (relational, network or hierarchical) that does not equate to both having the same datatypes between them. For example, Oracle and Microsoft Access are both relational databases, though each has its own unique datatypes. Access has a "MONEY" datatype, but Oracle does not. Oracle does have a numeric type "NUMBER(,#,#)" that can be transformed into an equivalent form of the Access money datatype by means of XSLT transformations. Hina

(2000) and Eddie (2001) both addressed similar problem areas of databases and repository data sharing within DoD using XML at the application level.

Lawler (2003) took an object-oriented approach to data interoperability by combined XML's data binding and Java's object-oriented design. His method is to write application objects to an XML document, then transmitting the document over a network to a destination, which then converts the XML document back into Java objects. This method removes the system-to-system problems of disjoint datatypes and structures, enabling heterogeneous applications or systems to communicate and share their data given the data (objects) are inside of XML. This is possible because XML is text, which is platform independent.

2. NPS Tract Data Conversion Suite (TDCS)

There are a numerous C4I data languages that define tactical information within DoD for air, land, sea and undersea contacts, but each defines its data differently preventing data interchange. The Naval Postgraduate School in conjunction with the Naval Undersea Warfare Center (NUWC) has reviewed the data formats from many C4I systems and has found that while format is different between systems, often the data is saying similar things, specifically, tract data such as time, position, identification, velocity and sometimes area of uncertainty (AOU) estimates. Additionally, most of these system's recorded their data in and XML format or a format that is convertible to XML.

Using an XSLT philosophy as just discussed, NPS is arguing the thesis that track data within each of the systems is the key to enable interoperability and processing of multiple data formats by means of data transformation to a common data format, summarized in Figure 7.

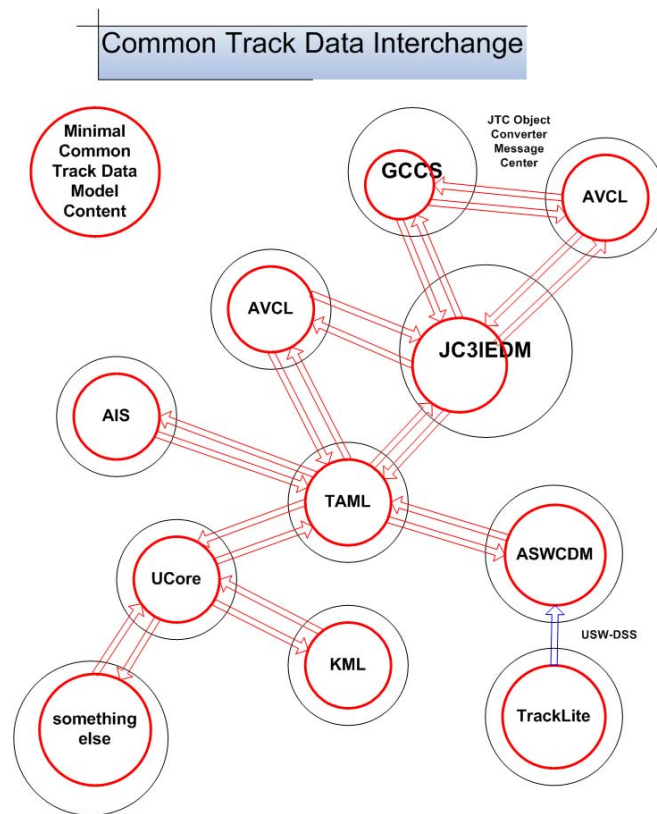


Figure 7. C4I Systems with Common Tract Data Interchanges

To achieve system-to-system interoperability, NPS and NUWC is developing a Track Data Conversion Suite (TDCS) that transforms each system's tact data to a common data format that is understood by all. They are addressing the issue by adding Semantic Web rules to express common data using Resource Description Framework (RDF), Web Ontology Language (OWL) and Semantic Web Rule Language (SWRL) to define data relationships, Figure 8.

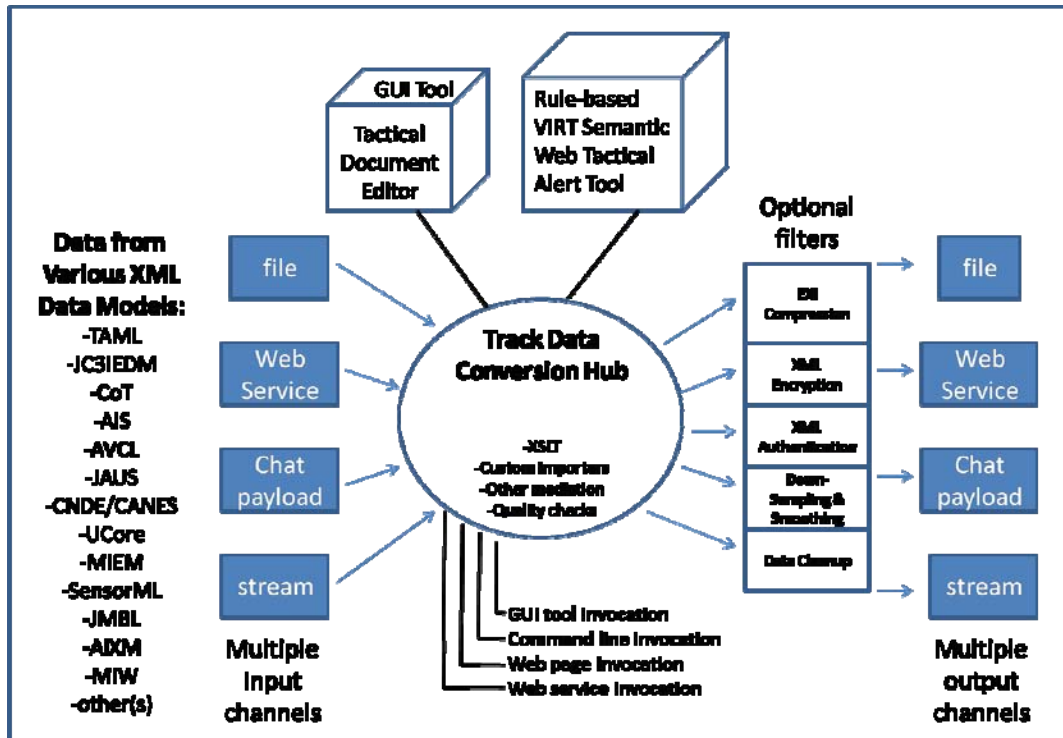


Figure 8. Tract Data Conversion Suite Conceptual Architecture

This Tract Data Conversion Suite philosophy of common track content presents an opportunity for information interchange. TDCS is not trying to map all parts of all languages to every other, rather it is simply concentrating on converting track data. This approach is likely to maximize effectiveness among large cooperating sets of C4I systems. Having a common basis in XML simplifies these conversion tasks

The TDCS assets are open source and maintained under version control available at <https://savage.nps.edu/svn/nps/TrackDataConversionSuite>. Current work includes adding further language conversions, and building a test corpus of example documents for automated testing of validity and coverage. Usage and participation by U.S. government agencies or contractors is welcome.

3. Semantic Web

Data interchange is reasonably well understood and manageable for system-to-system data repositories, although it does require programmers with detailed knowledge of both sending and receiving applications or systems to develop the algorithms to do the transformations; transformations that are essentially static and permanent. While XML is a great universal middleware to enable data interchange, it does nothing organically to eliminate the ambiguity of data meaning or intent. During data transformation, the human understood relations and correlations between data are not captured, only the raw fact that a piece of data is numeric or alphabetic. For the rudimentary data exchange this is all that is needed. However, on the battlefield decisions need to be based on the understanding of the data and how all combinations of data are related. There are just too many ways to represent a number or a text-string, moreover, too many possible meanings for any data in raw format. For example, both “cat” and “kitty-cat” are the same thing to a human, but raw XML views each as being distinctly different without any meaningful relation or correlation. Assuming hundreds of thousands of uncorrelated cat and kitty-cat instances, decision makers quickly enter an information-overload situation. That is, the information is present, but the useful meaning of the information is lost in the clutter. The Semantic Web presents some relief to such information overload by meaningfully relating the raw data facts through ontologies. A simple analogy of this concept is to think of the Semantic Web as an intelligent thesaurus that can define how arbitrary data elements are similar: cat is equal to kitty-cat (or perhaps slightly different, depending on context).

Tactical data exchange networks that share situational information between commanders within DoD often result in information-overload situations. The traditional methods to find important information for a particular commander is to station a human to read and summarize all incoming messages, or to automate the same process by hard-coding keywords into a message parser that flags messages containing keywords. Neither of these solutions is perfect. Humans can err, and automated methods might not err in controlled circumstances, though they also will not deviate from the assigned keywords even if the situation context or information meaning changes.

Childers (2006) explored this problem and found XML to be a ready-made format to share semantic data. Her approach was to show the effectiveness of Semantic Web ontologies for exchanging network's data using XML. Semantic Web languages such as Resource Definition Framework (RDF) and Web Ontology Language (OWL) provide a well-known and supported framework to share and distribute relations about the raw data: hostile is the same as red is the same as enemy.

A domain with challenges similar to semantic correlations between tactical data-exchange networks is Knowledge Management (KM) systems. Here, a massive amount of information is in the system, but finding the right information within the system is difficult; it is often hard to find relevant information due to poor names, relations or locations. KM systems are only as good as the administrators maintaining the system and the quality of the processes guiding information contributors; what is meaningful to one person is not always the same case for another person. McCarthy (2004), like Childers, found that the Semantic Web concept as a solution to this problem, and that XML is the ideal vehicle to distribute and store KM information coherently.

An exemplar case of successful use of XML as a KM structure is the U.S. Coast Guard's Port Information Knowledge Management System (Stewart, 2003). The system captures After Action Reports (AAR) of port visits in a standard Web-enabled client-server architecture that enables the insert, update and review of port-specific information. This on-request system provides a ship an advanced look of what to expect during a port visit prior to arrival based on previous ships' experiences. These experiences are then war-gamed and simulated for the optimal port security configuration during the port visit.

The conclusion is that data must be interoperable between heterogeneous systems in order to build a good tactical system, but just sharing the information is not enough. The next generation of systems must also be able to understand the data well enough to infer deeper meanings, and be able to present the information to the decision makers within the context of the current tactical scope. XML is the ideal vehicle to be the link between heterogeneous systems' data, and is the needed solution to extend the powers of the Semantic Web to the end-users.

4. Network-Centric / Force-Net / Future Vision

The visions of future DoD network systems are being built off of the Semantic Web concepts (DOD CIO IM, 2006). Future system focus is on Web-enabled networks of people, processes and technology to empower and provide decision makers with accurate, ubiquitous and timely information, as they need it in order to make optimal decisions. The result is a system-of-systems where every sailor and soldier is a sensor. The secondary aspect of the vision is to make network systems universally adaptable, not focusing on any one platform's or location's unique abilities or context. In order to leverage the masses of data for the common good, the data must be presented within a meaningful framework that is not of any one proprietary design to ensure integration (Reynolds, 2006; Neushul, 2003). Data from the disjoint and dissimilar systems must be able to integrate data meaningfully and seamlessly between legacy and the next generation of systems (Hodges, 2004; Denny & Jahn, 2005). Both DoD and DON CIO have chosen the SOA architecture as the framework, utilizing the XML family as the implementation mechanism (DON CIO, 2006; DoD CIO, 2007). This guidance for Implementing Network-Centric data sharing is summarized in Table 2 (DOD CIO IM, 2006).

Data Effect	Scope Of Enterprise Role	Scope of Community of Interest Role
Make Data Visible	<ul style="list-style-type: none"> • Develop, maintain DoD Discovery Metadata Specification (DDMS) to facilitate DoD-wide search • Direct development of Enterprise search capability 	<ul style="list-style-type: none"> • Tag data holdings with DDMS • Extend for COI specific search criteria
Make Data Accessible	<ul style="list-style-type: none"> • Maintain repository of acceptable commercial standards for Web-based services • Direct development of federated service registry for Web services 	<ul style="list-style-type: none"> • Implement access services • Register access services in federated service registry
Make Data Understandable	<ul style="list-style-type: none"> • Direct development of federated metadata registry for semantic and structural metadata 	<ul style="list-style-type: none"> • Develop vocabularies, taxonomies for data exchange • Register these agreements in federated DoD metadata Registry

Table 2. Network-centric Notional Community of Interest (COI) Data-Sharing Roles (From DOD CIO IM, 2006)

5. Network Security Considerations

Future systems will provide services for their applications within Web-like environments. Similar to all other information systems, Web-based systems have network security problems that produce interoperability limits. For example, assuming a front-end proxy server is not used, unique ports are typically required for each application that must be open along the entire path from the server to the client of the communication link. However, this is not a reasonable assumption for future systems' development. Good security practices limit the number of open ports to an absolute minimum,

permitting only essential applications direct access to the external network. This port limiting action might inhibit the development of Web-like systems since each Web-like application requires its own unique communication port; a port likely not to be open. However, through the Simple Object Access Protocol (SOAP), which is a HTTP standard port 80 XML-based messaging protocol, can be worked within this port limiting security practice (Rosetti, 2004). The solution is to do all communication by means of SOAP HTTP wrappers over port 80 or 443; the HTTP port and likely always open. The key hurdle to this type of solution then becomes the ability of the HTTP server to distinguish between HTTP and SOAP services, and then to branch processing accordingly.

6. Tactical Chat

Today's battlefronts are monitored and supported by means of chat-rooms in all Areas of Operations (AOR). Tactical chat has proven to be an invaluable tool within all warfare domains. A quick review of any AOR's Operational Tasking (OPTASK) or Operational Order (OPORD) shows that all warfare commanders have establish at least one tactical chat channel to maintain real-time communications, and most AOR's have establish a dedicated OPTASK Chat. Tactical chat is, and will remain the preferred tool for battle commanders to fight their battles in terms of real-time fully distributed tactical communications.

Reminiscent of the database interoperability problems presented at the beginning of this section, many chat applications employ proprietary protocols that do not interoperate with other chat applications. This lack of interoperability requires commanders to coordinate specific chat applications within their subordinates prior to entering an AOR. Without this coordination, there is no assurance of communication ability via chat. This coordination is trivial if the force structure remains the same from the beginning to end of an operations; the commander simply issues an order prior to starting an operation for all subordinates to install the designated chat application. However, if a new command such, as allied country joins an in progress operation they might be excluded from the preferred communications channel, chat, if they do not already have the pre-established chat application. Because chat communications have

become the dominant information distribution method among multiple participants in real-time, those subordinate commands without chat are at a disadvantage.

DeVos (2007) and Arnold (2006) introduce XML Tactical Chat (XTC) based on the XML-based Extensible Message and Presence Protocol (XMPP) as a non-proprietary XML-based Chat application. Because XMPP is XML-based, it is Web ready by default and in line with the Network-Centric SOA architecture. Because it is XML-based, it can encapsulate almost any other form of data within it such as the IEEE XML encoding of Distributed Interactive Simulation (DIS). Since it is not proprietary, anyone can use XMPP freely ensuring mass distribution simplicity. Since it can transmit other than text data, XMPP becomes a multi-purpose application that can grow as the future battle needs grow. The Defense Information Systems Agency (DISA), the managers of the DoD networks, requires the use of the XMPP rather than proprietary or closed chat protocols (Jabber, 2006).

7. Modeling and Simulation (M&S)

Essential for the next generation of war fighters is the ability to simulate intended battles and actions prior to execution. Automatic feedback generated from a simulation is commonly referred to as an After Action Review/Report (AAR), and generally represents AAR data within an XML format. An automated AAR serves two important purposes: 1) reduces the man-hours intensive simulation performance evaluation process; 2) provides follow-on training opportunities such as simulation playback in both textual and or visual formats.

While man-hour reductions are important, the real value of AAR is the playback ability. The playback of an AAR from various viewpoints or contingencies enables decision makers the ability to refine their intending decision by learning new insights into a problem that they may not have realized prior to the use of an alternative simulation viewpoint. Filiagos (2004) demonstrated the capability of capturing AAR in an XML format for 3D visualization. This enabled users to review their actions after the fact in 3D format from any vantage point, empowering the simulation end-users to grow from their simulation experiences in ways never before been realized.

A benefit of AAR's is that HPC simulators cannot be installed at every location to run simulation due to size and cost, although playback tools can. AAR are easily shared remotely around the world, allowing large numbers of participants and reviewers, ensuring the optimal preparations have been made for any intending battle or action. The concept flow is a centralized location houses an HPC, executes a simulation and then share the AAR with the masses. This ensures all users have a consistent and matching understanding of the simulation results or implications. Additionally, users with different purposes or scopes are able to use the same simulation AAR, but from different viewpoints to gain their unique scoped insight desires. In the pursuit of a distributed simulation like concept, Rashid (2009) describes a candidate network architecture that is XML-based integrating a Massive Multiplayer Online Game (MMOG) server.

The use of simulations is nothing new, all branches of DoD uses simulations in varying way to solve problems and train personnel. Sadly, there is no standard format for simulation repositories or state information; denying the sharing of information between simulators. As an example, the Navy and Army both conduct simulations on basic ballistics, though possibly using different simulators. Ballistics is a relatively simple and understood concept, but because the Army and Navy may be using different simulators, their results are not interoperable between the services due to simulation data differences. Hout (2003) addressed this simulation data interoperability problem by recommending the use of an XML-based data-structure. He noted simulation systems produce an output, but the output can only be interpreted within the same simulation system type. Similar to the problem and resolution of data interoperability and tactical chat presented earlier in this chapter, XML is the ready-made solution to the simulation system data interchange, XML by means of XSLT to act as a middleware between simulations. All simulators are reasonably the same, differing primarily in naming conventions which XSLT mapping can overcome.

Modern war-fighting machines include unmanned vehicles, with the most advanced being able to operate autonomously. These complex machines often operate cooperatively toward a common goal of feeding information to and from one autonomous vehicle to another without user involvement. In order to prove and evaluate the

effectiveness, accuracy and lethality of these techniques, they must be simulated. Davis (2006) points out, current autonomous vehicle interoperability is limited by vehicle-specific data formats and support systems. He presents an XML-based solution using the Autonomous Vehicle Command Language (AVCL) for the translation between data formats enabling simulation and execution interoperability, with XML as the middleware.

8. Internet and Communications Security

As addressed in Securing Cyberspace for the 44th President (Langevin et al., 2008), Internet security is a key focus as is the expanding of the Web. Executive Order (EO) 13356 called for improved sharing of terrorist information to protect Americans. The Force-Net and Network-Centric vision sets the tone for capabilities-based service infrastructure for ubiquitous access to timely, secure, decision-quality information by edge users (ASD NII, 2006). Each of these directives boil down to visions of Web-like data sharing among decision makers, where decision makers could be Joint operations or coalitions partners, and the data semantics are the same regardless of the user. Simply, every decision maker has the information they need, and that information is the same regardless what domain they are working within limited only by security clearance.

These visions present security considerations not present within legacy stovepipe systems. Stovepipe systems are secure due to the fact that only those authorized to have access to them, have physical access. Additionally, stovepipe systems do not follow a well-known format, often achieving only the appearance of security through obscurity. They are secure by architecture, but that crude form of security makes them unable to interoperate between heterogeneous forces and systems, including other stovepipes, and unable to keep pace with evolving information needs.

Focusing on a Web-like solution presents interoperability improvements, but also introduces new data security control problems. The sharing of data from a unified data repository jointly between multiple security levels introduces the risk of sensitive data leaking into unauthorized security domains. The potential for classified data being shared with those who do not have appropriate security levels is increased as data becomes more distributed, e.g., Web-based.

A classic solution for Web-based environment security is the use of Keynote policies, which is the tagging of files with access authorization rules (Keynote, 1999). These policies are created, and assigned to each file on a server. They are defined in text form delineating who, when, what and various other conditions to apply to a particular file or directory when read or write request arrive. When a user request access to the file, depending on that user's rights, access is granted or denied. Mohan (2002) pointed out that the style of the Keynote definitions are essentially tagged information like XML, but are of a Keynote unique format. He showed that Keynote could easily be mapped to an XML format that provides a more reliable, user-friendly and transportable solution to file-server security considerations.

Taking the work of Mohan and placing it within the Network-Centric vision of a Web environment, Kane (2005) showed that a Trusted Computing Environment (TCE) of organized and controlled data could establish server-based security. His used XML as the packaging for data based on the user's permissions, single person or group, and presented the data to a Web interface.

A caveat to both the TCE and Keynote methods is they rely on users to have access to the repository. This direct access might not be possible or realistic due to security restrictions or inability to connect to the repository server. The tactical environment of war often does not permit internet access, although data-access needs remain.

Estlund (2006) presented an XML element-based security approach. Requested information placed in an XML file at all security levels, with each security level represented as a new element with equivalent security access control placed on its contents. What this provides is a single document with all data from unclassified to higher security classification, and when opened, only reveals that user's or domain's authorized content; the same information is available to all recipients. This single file can be manually delivered to locations without access. However, this method comes with the dependency that multi-level security applications are the only way to retrieve the data and ensure proper dissemination.

Williams (2009) presented a XML document level security approach by using common digital signature and encryption that enables the transmission of classified data across unsecure networks. This is practical in that XML is Web service orientated, and that digital signature and encryption are robust techniques available on most networks. Baring a lack of network access, this approach supports open network transmissions of classified data, and can be tailored to meet any dynamic needs of warfare, such as dealing with the disparity in security access rights of coalition forces.

E. CHAPTER CONCLUSION

XML is everywhere, and though it is a text only format, it can include other data formats by converting from binary to text using base-64 or other encoding. XML is mandated for use within the DoD, and it is the default standard for everything IT in both the business world and DoD. XML is, and will remain a major player in every IT advancement in the future. Decision about the future are being made today with XML as the backbone architecture of choice because XML presents a ready-made, proven, reliable, and understandable solution to interoperability, as well as simplified architecture for rapid development.

F. CHAPTER SUMMARY

This chapter introduces the XML structured format and lists the breadth of XML penetration within both DoD and the business world. Examples of XML usage from both areas are highlighted by examples.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SHORTFALLS OF XML LEADING TO EXI

A. INTRODUCTION

This chapter discusses the shortfalls of XML's design structure, verbosity and processor-intensive parsing that place limits on the network deployability that native XML 1.x can reach, specifically to handheld, mobile and wireless devices.

B. FILE SIZE

1. XML is Verbose

XML is verbose, meaning XML source text has a lot more than just the information contained within the XML document. Put another way, the unique information contained within an XML document makes up only a portion of the document's size. Looking at the simple notebook.xml example, repeated in Table 3 below for clarity, only the values from attribute content, the characters surrounded by quotations, or element content, the characters between the beginning and ending element declarations, are actual unique information of the document. All other characters are XML structural, metadata, which loosely means information about information, i.e., the verbosity of XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<notebook date="2007-09-12">
  <note category="EXI" date="2007-07-23">
    <subject>EXI</subject>
    <body>Do not forget it!</body>
  </note>
  <note date="2007-09-12">
    <subject>shopping list</subject>
    <body>milk, honey</body>
  </note>
</notebook>
```

Table 3. Notebook.xml Verbose Structured Format (From W3C, 2007)

Stripping all of the XML structure from the raw notebook.xml document and leaving only the actual information of the document, Table 4 is generated.

2007-09-12EXI2007-07-23EXIDo not forget it!2007-09-12shopping listmilk, honey
--

Table 4. Notebook.xml Terse Information-Only Format

The original verbose version of the notebook.xml in Table 3 has 310 characters including spaces, and the information-only version in Table 4 has only 77 characters. Only 24% of the size of the document is data, the remaining 76% is XML formatting and structuring characters.

2. Why XML is Verbose

The fundamental problem with the information-only version of the notebook.xml document is it is nearly impossible to determine the meaning of the information; there is no clear way to determine where one piece of information starts and ends, nor how the pieces of information relate to one another. Without a detailed pre-existing knowledge of exactly how many pieces of information are contained in the document, or knowing exactly how long the number of characters used by each piece of information, extracting the individual attribute values and element contents from the terse information-only version is difficult to impossible to perform. For the purposes of general XML, this information-only approach is clearly impossible. Even knowing the meaning of the notebook.xml example it is still hard to decipher the information when looking at the information-only version.

Prior to the advent of XML, this problem of information representation was addressed with a file-structure, which is essentially XML structure hardcoded into a procedural programming language within an algorithm to parse the information from a file. This requires the parsing algorithm to know, by means of hardcoded procedural routines, the precise number of data field and the exact number of characters of each data field within the file-structure (Stern & Stern, 1994). Note that a field is a piece of

information from a list, or properly called a record. Using the notebook.xml example, the subject element is a field of the record note and the notebook.xml contains two note records.

The problem with this approach is that it mandates a rigid input file format; a format that cannot change. The file has to consist of the same number of fields with the same number of characters every time, and is unable to support new fields or varying length fields. The traditional file-structure paradigms cannot adapt to different file formats or slight deviations without source-code modification followed by recompilation. File-structures just cannot keep up with the dynamics of the information-based world of today.

3. XML is Verbose by Design and is Not New

XML's flexibility to support an arbitrary information set requires that it declare every element and attribute explicitly every time, even if the element or attribute is equal to a previous element or attribute in order to show the parent-child relationships clearly and unambiguously. Overall, this equates to a verbose XML document, although it is one of the design tradeoffs considered during the XML development, design point four "XML is Verbose by Design" from the W3C *XML in 10 Points* document (1999).

It can be reasoned that XML's structured format is not extra wasteful information, since XML's verbose structure defines the information of a document the same any other file-structure does with strict structural hard-coded algorithms. XML resolves the strict structural issues of file-structures, but at the cost of being verbose. The W3C *XML in 10 Points* documents note 6, "XML is new but not that new," points that XML is the evolution of pervious data-structuring formats such as SGML (W3C, 2003). XML is really no different than a file-structure, but unlike the classical sense of a file-structure, XML is not hardware driven nor is it driven by any particular data format, unless explicitly stated so by means of a schema, and so, it is able to flexibly define any information for nearly any purpose. XML allows a document writer to define what elements and attributes are to be in a document without specifying any particular information about the information itself, unless explicitly by means of a XML schema.

XML has no rigid structuring requirements of the information itself other than information tagging, which leads to the verbosity of XML.

Another simple example of the XML size implications is Table 5. This XML document (top), when processed by a browser consists of a single 3D box (bottom).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.2//EN"
    "http://www.web3d.org/specifications/x3d-3.2.dtd">

<X3D profile='Immersive' version='3.2'
    xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
    xsd:noNamespaceSchemaLocation=
        'http://www.web3d.org/specifications/x3d-3.2.xsd'>
  <head>
  </head>
  <Scene>
    <Shape>
      <Box/>
    </Shape>
  </Scene>
</X3D>
```

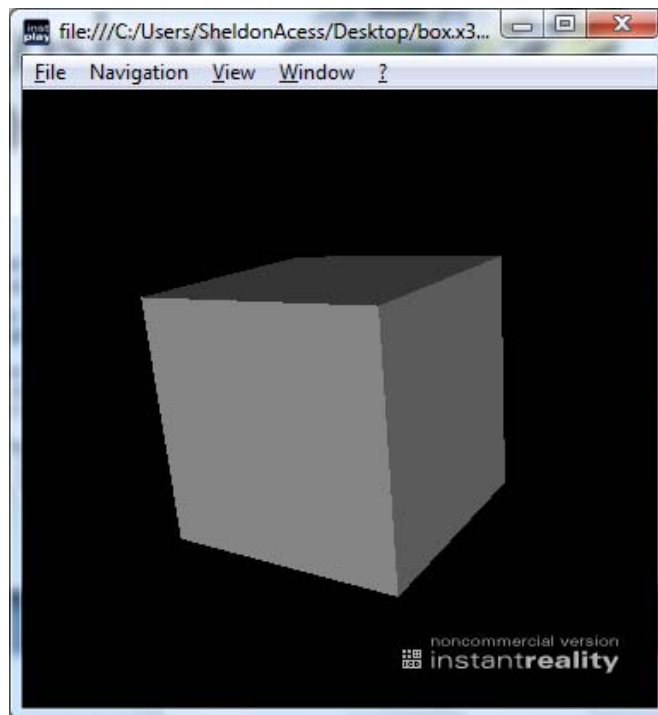


Table 5. XML Structure Bloating Example
(X3D XML Code Top, X3D Scene Bottom)

This Extensible 3D Graphics (X3D) document, which is a member of the XML family of languages, consist of 355 characters, but only represents a box, defined by three characters, the bold “Box” statement code at the top. Everything else in some sense is overhead metadata information. This of course assumes that there is a language vocabulary that defined what a box is, its location, colors, etc, which in this example is the case, all based on the X3D specification.

In summary, when discounting structural semantics, the sizes of XML documents are larger than the actual information contained within them due to XML’s intentionally verbose design.

C. PROCESSOR INTENSIVE

1. Legacy File-structure Formatted Data

The file-structures of the past, in addition to requiring a data size declaration, also required datatype declaration. If the data field is numerical, it must be declared as numeric including the maximum number of digits the value might ever contain (Stern & Stern, 1994). If the data is to be a date, it has to be declared as a dated datatype, which usually for data datatypes equates to a strict platform-dependent data format. This declaration method repeats for every data field within each record. For file-structures, this became a burden for several reasons:

- Inputs are often in an incorrect format, data field sizes change, and all record might not contain all the fields.
- Much of the processing of an input file is dedicated to error checking and correction, and not the processing of the actual data.
- Endian disparity (determination of the most significant bit within each byte of information) between different CPUs (Blanc & Maaraoui, 2010).

2. XML's Resolution to the File-Structure Problems

One of the methods employed by XML to resolve the problems of historical file-structure is to eliminate the problematic numerical and other platform-dependent datatypes by representing everything as plain text, Unicode; design point three “XML is Text, But Isn't Meant to be Read” from the W3C (1999) *XML in 10 Points* document.

This design consideration eliminates the big and little endian disparity between different CPUs because Unicode text has a standardized bit pattern between all CPUs, unlike traditional numeric datatypes (Blanc & Maaraoui, 2010; Unicode, 2010). This also allows XML to be extended to support any arbitrary datatype since each piece of information is tagged indicating its start and end without requiring a predefined datatype or sizes, and does so in a CPU-independent format.

XML's structured format also supports modifications to existing information, such as allowing missing or new fields without crashing an application processing an XML document.

Additionally, through the usage of XML, the previous input error-checking processes required for legacy file-structures can be relaxed, making XML file processing more efficient, adaptive and robust. Thus, the high reliability of data expressed as XML can greatly reduce the size, complexity and maintainability of software programs.

3. Processor Intensity String-to-Number Conversion

The cost of representing all data as text impacts numeric information during the string-to-number conversion process. For numerically heavy XML documents, the text-only approach becomes a computational burden on the CPU. The conceptually simple activity of string-to-numeric conversions, sometimes called cast parsing, requires many CPU cycles, making cast parsing a time-and-CPU taxing process. There are many algorithms designed to perform these conversions with varying levels of complexity, though they all follow roughly the same execution flow as listed in Table 6.

```

Declare a number buffer set to zero

For each character to the left of decimal at count i [0, length)
    Find character's 0-9 numerical equivalent
    Set base multiplier as 10^i
    Multiply by base multiplier
    Add to buffer

For each character to the right of decimal at count i [0, length)
    Find character's 0-9 numerical equivalent
    Set base multiplier as 10^i
    Multiply by 1/(10 * base multiplier)
    Add to buffer

Return buffer

```

Table 6. String-to-Numeric Conversion Algorithm

This sequence of operations must be performed on every numeric value within an XML document before its value can be used by the processing application with numeric intent. For a single digit, this cast parsing process requires the declaration of storage space for the temporary buffer structure; a table lookup of worst-case nine conditional statements, a multiplication followed by an addition. This means for every numeric character, 11 discrete computations must be performed before the actual intended numerical value from the XML document can be used for its intended purpose, which then must be also further processed by the application using the XML document.

Numeric values XML documents increase the initial processing overhead of XML that can be beyond the capacity of small devices.

4. Processor Intensive Searching

Aside from the processing cost associated with XML's text-based format, XML also does not have efficient indexing or searching techniques. XML documents are loosely defined, even with a supporting schema definition, so there is no form of efficient data indexing or searching. With the lack of deterministic structure, XML document searching is limited to a linear approach, which is the slowest search algorithm.

D. IMPACT ON HANDHELD DEVICES

1. Bandwidth Limitations

The example presented in Table 5 is a simple document of a single box object, but as the number of objects increases, so does the associated XML structuring and ultimately the XML file size. The file size of an XML document quickly becomes an issue when it is faced with the Network-Centric vision to push the internet deeper down range, and into the hands of individual sailor and soldier handheld devices or to remote fighting units. These devices and remote units generally do not have bandwidth support beyond that of a typical telephone dialup speed at best. The larger a file is, the longer it takes to send or receive on low-bandwidth devices, if able to transmit at all. Large XML means limits on the depth into the battlefield that information can be pushed using current connectivity.

Putting a dialup speed into perspective, Figure 9, adopted from Popovich (2005), describes the transfer times of a notional military file against a number of different bandwidth capabilities, assuming 100% dedicated bandwidth utilization.

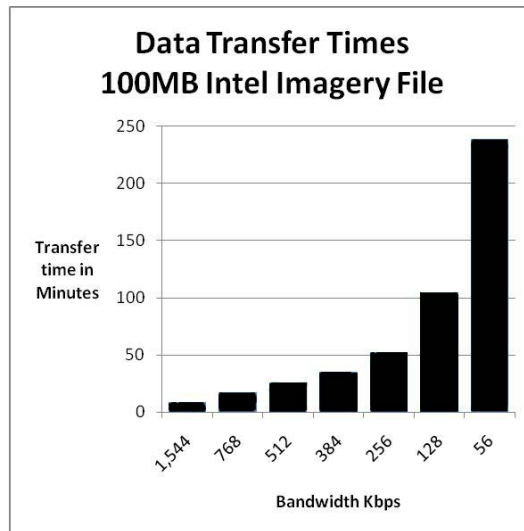


Figure 9. Example of Transfer Time vs. Bandwidth (Data Rate) for a 100MB Video or Imagery File (From Popovich, 2005)

Military relevant files in XML format can quickly reach the multi-megabytes in size resulting in remote units unable to meet the bandwidth requirements to send or receive the XML format even though the fractional information within the XML document is within the limits of the remote unit's transfer abilities.

2. Battery and Heat Limitations

To push networks deeper into the battlefield, handheld devices, similar to digital organizers, are the devices that are most likely to enable the push. These devices generally come with full-feature color interfaces that have a look-and-feel similar to an interface found on a desktop computer. They have through functionality but with specialized Application Programming Interfaces (API) designed specifically for their processors and display abilities. By matching a creative architecture design with a handheld API based on a few subtle limitations, handheld devices do almost all that a traditional desktop can, but within the palm of your hand.

Handheld processor limits are constrained by many things, from the number of operations per time-cycle, to the maximum storage ability, to the amount of power it requires to operate. The key things that set handheld devices apart from a desktop computer are power consumption, heat dissipation and endurance. A desktop computer is connected to a constant power source, while a handheld device relies on the power available in the attached battery which has a finite capacity.

a. Battery Power

A fundamental problem for handheld devices that is not present for desktop computers, at least not to the same level of importance, is a measurement of power consumption due to processor operations. Notionally, a desktop computer is evaluated on the number of operations it can perform in a specific time period, CPU clock-speed, along with its memory capacity, without consideration of power consumption given that a traditional desktop computer is connected to a constant power supply, wall outlet. On the other hand, handheld devices operate on a limited power supply, i.e., their installed battery, and must take power consumption into consideration

for processor operations. A desktop if confronted by a complex file has essentially unlimited electrical power to throw at the problem. In contrast, the more complex a file a handheld device has to process, the faster that file drains the handheld device's battery. This reasonably simple to understand concept is often overlooked, and results in the premature demise of many handheld device applications because the application requires too much power to effectively operate.

The immediately hoped-for solution to this problem is to make better batteries. Moore's law states that computer abilities double every 18 months. However, battery technology has not kept pace with Moore's law, and the science behind battery technology indicates current technology may already be at its peak capacity (Ultra, n.d.). Handheld devices for all intended purpose are nearly equivalent to desktop computers in terms of processing ability, but are limited unequally on electrical power. A powerful handheld device can quickly drain its battery and become useless on the battlefield. This is a potentially deadly situation if the devices are performing key mission functions. Similarly, if a handheld processor is not able to perform the complicated tasks required of a battlefield mission, it becomes useless even before it can reach the battlefield.

b. Heat Dissipation

Compounding the power problem of handheld devices is heat dissipation. A desktop computer has a fan to move air across the processor to cool the motherboard and other heat producing parts within the computer shell. Handheld devices have no fan to move air, relying only on natural airflow. Handheld devices are "held" while operating, no air flows around the device to cool it. The designed operation of handheld device places them in a dangerous position to meltdown.

Handheld devices cannot dissipate heat at the same rate as a desktop computer so in addition to being limited by power, they are also limited by their rate of heat transfer, which is a function of power. The faster the battery is drained, the more heat it produces, the more heat that is produced, the faster the battery is drained.

E. CHAPTER CONCLUSION

While XML is the defacto data format for information exchange, its vices, particularly when being processed on handheld devices, can make XML too large and complex to be efficiently process. Due to XML's verbosity, it prevents low-bandwidth devices from being able to receive and transmit XML documents. Due to the text-only format of XML, small devices are unable to process largely numeric XML documents due to the complexity of numerous text-to-number conversions. Complexity is further confounded within small devices by limited battery life, limited memory and small CPUs.

Handheld devices while nearly as functional as a desktop are limited by how long they can operate due to power and heat. The more complex an operation, the more power a handheld device will need. The more power a handheld device is using, the more heat it is producing. The more heat a handheld device produces, the faster it drains the battery.

Ultimately, the constraints of XML limit the network depth to which XML can be deployed due to the same design structure that has delivered XML's successes.

F. CHAPTER SUMMARY

This chapter highlights the verbosity and complexity of XML processing that can prevent handheld and other small devices from being able to adequately process XML documents, limiting the network depth of XML.

THIS PAGE INTENTIONALLY LEFT BLANK

V. BINARY XML FORMAT RATIONALE: XBC

A. INTRODUCTION

This chapter discusses the history of the awareness of XML's verbosity and parsing problems. Cases from DoD and the business world's efforts to alleviate the limitations of XML are discussed. Based on the findings of the XML Binary Characterization (XBC) working group, the overall purpose of this chapter is to discuss the technical challenges that an alternative XML format solution faces, and to discuss some of the general metrics to ensure that a successful alternative XML format is achieved.

B. XML BINARY CHARACTERIZATION (XBC) WORKING GROUP

The primary rationale for a compressed binary XML representation is to grow the Web by enabling XML to serve use cases that otherwise are unable to support native XML 1.x. The W3C's (2005) XML Binary Characterization (XBC) working group discusses the feasibility of an efficient compressed binary-like representation of the XML Infoset within the XML industries. The XBC started with the simple goal of finding a universal method to efficiently serialize XML in an attempt to resolve its problematic issues: verbosity and efficiency.

The limitations of XML are nothing new. Almost since the introduction of XML, various independent groups have been trying to optimize XML. One of the earliest attempts was in 1999 by Siemens, but due to lack of interest in the early onset of XML for compression, their efforts were not adopted (Siemens, 2003). However, now that XML's worldwide footprint is enormous, industry-wide concern and attention has been refocused on XML.

As introduced in the previous chapters, XML is found in nearly anything and everything digital today. This simply equates to more and more data being encompassed into an XML format. As time progresses, the next generation of XML data will continue

to increase in size and will encompass more datatypes than traditional text to include multimedia, Modeling and Simulation (M&S), and tactical or data-streaming channels. Due to XML's own success of being the preferred format for data representation and distribution, with these new datatypes, the limitations of XML will be exacerbated. Therefore, the next logical avenue for XML is to find an alternative method to reduce its size and complexity in order to ensure its continued growth and success. In other words, an alternative compact and efficient XML format is needed.

The XBC group set several design rules in their pursuit of an alternative XML format: it must meet the needs of all existing XML family of languages, be backwards compatible, and it cannot exclude any member of the XML family of languages (W3C, 2005). The XBC further compiled a list, Table 7, of common domain use cases that are at risk of being unable to support native XML 1.x within their *XML Binary Characterization Use Cases* document (W3C, 2005). This document also lists, as shown in Figure 10, the unique demanded properties that any alternative XML format will have to support.

1. Broadcast Systems
2. Energy Industry (Floating Point)
3. X3D Graphics
4. Web services Small Devices
5. Web services Enterprise
6. Electronic Documentation
7. Security Industry
8. Multimedia
9. Inter Business Communications
10. XMPP Chat
11. Data Persistent Storage
12. Business and Knowledge Processing
13. Routing and Publishing Subscribe
14. Web services Routing
15. Military Information Interoperability
16. Sensor Processing and Communication
17. Data Synchronization
18. Supercomputing and Grid Processing

Table 7. W3C List of Domains That are Unsupported by the Native XML 1.x Format (From W3C, 2005)

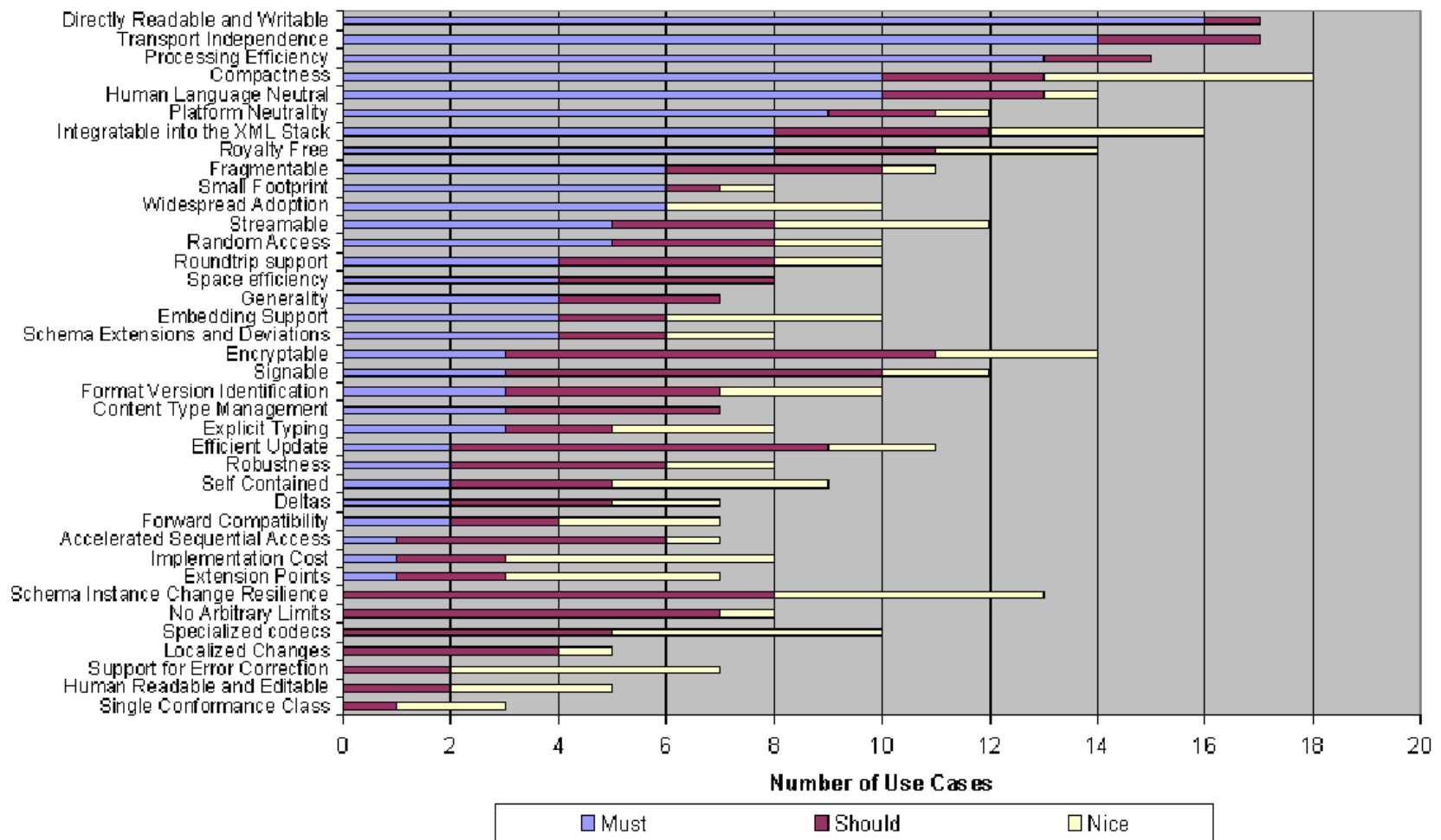


Figure 10. Binary XML Format Design Property Demands (From W3C, 2005)

The XBC working group consensus on the need for a robust, compatible, efficient XML format that supports static cases, streaming cases, real-time processing, and is forward technology aware. All of the greatest aspects of XML must be maintained, although retaining the text format remained unlikely. Overall, make XML small, fast, efficient, universally adaptable, don't add complexity, and ensuring the legacy of XML in the wild remains unaffected.

C. DOD INTERESTS IN AN ALTERNATIVE XML FORMAT

Since the ASD (2007) mandated the use of XML, a number of analyses of the effects of XML's usage within the DoD have been conducted. One such study was conducted by the MITRE Corporation (2008) for the Army and Marine Corps to address the file size of XML under low-bandwidth conditions, and in support of legacy stovepipe messaging systems transition to an XML format.

For the MITRE study, three XML-based military messages were generated: two Operations Orders (OPORD) with one of medium size and the other large in size relative to actual message sizes in operational use, and one Position Report. The study then applied three different XML vocabularies (PASS, VMF_XML, FBCB2) to the messages, except no FBCB2 schema was available for the position report. These schema-informed XML messages are then compressed with traditional Zip and the W3C proposed alternative XML format, EXI, as a preliminary look at alternative XML formats for tactical use. Their results are listed in Table 8.

Message Format	POS REP	OP ORDER 1	OP ORDER 2
PASS XML Message (Original Size)	1144	5,526	17,507
% of Original (PASS Zipped)	58%	34%	21%
% of Original (PASS Efficient)	6%	25%	21%
VMF_XML XML Message	1394	8,557	25,304
% of Original (VMF_XML Zipped)	47%	24%	16%
% of Original (VMF_XML Efficient)	6%	17%	16%
FBCB2 XML Message	NA	7,762	20,507
% of Original (FBCB2 Zipped)	N/A	28%	20%
% of Original (FBCB2 Efficient)	N/A	17%	19%

Table 8. MITRE XML Compression Comparisons Study on DoD Messages (After MITRE, 2008)

MITRE's summary of conclusions from their testing:

- Efficient XML should be considered as a messaging technology to be used, especially where large text-based messages are expected.
- Efficient XML reduces an XML document to 25% or less of the original document size.
- Efficient XML improves in efficiency as a message size grows.
- Efficient XML loses efficiency relative to other techniques for XML documents that contain binary data.

The MITRE Corporation (2003) conducted an earlier study for the United States Air Force evaluating solutions to mitigate the verbosity of XML using only Commercial Off-the-Shelf (COTS) solutions with intended implementation for low-bandwidth environments. Their approach was redundancy-based compression (WinZip and WBXML), schema-based optimization formatting (MPEG-7 and ASN.1), and a

hybrid of the two. Their results showed a 17:1 ratio of XML-to-binary average. While they did not conclude with a specific recommended solution, they did demonstrate that the Air Force could meet its desires to reduce the XML file sizes to a level compatible with low-bandwidth communications.

Norbraten (2004) studied the integration of DoD legacy stovepipe and modern information systems. He pointed out that the legacy systems operate in unpredictable and noisy domains, wireless and acoustic, which are highly susceptible to errors. His study evaluated how a binary XML format, the Extensible Schema-Based Compression (XSBC) can improve communication reliability within these noisy environments and achieve interoperability. Using a binary XML format on the outputs of legacy systems, classic error detection and correction techniques and compression can be used ensuring robust information exchange, and interoperability achieved consistent with the database interoperability work presented earlier in this thesis.

D. BUSINESS INTEREST IN AN ALTERNATIVE XML FORMAT

In 2003 the XBC working group solicited cases for and against a binary or other alternative XML format from the IT world. These cases were originally presented in the XBC's 2003 workshop and become the basis of the W3C's binary XML decision (W3C, 2005).

1. Arguments Supporting a Binary XML Format

Brutzman (2003), a professor at Naval Postgraduate School (NPS), argues for a binary XML format by countering the claiming that the GZip and other text-based compression techniques are good enough. He noted that superior XML-specific compression can be achieved if XML's structure is leveraged. Additionally, if an XML document has an associated schema, the datatype representation and overall documents structure can be used to create a substantially compressed, but equivalent document.

Cisco Inc (2003), an industry leader in network communication devices, heavily leverages XML and recognizes the importance of XML for Web services. While they

have not developed an optimization tool for XML themselves, they do see the need for efficient XML processing to take the Internet into intelligent networks: Web services, Semantic Web applications and others.

Siemens (2003) is working to develop an extension of the Web for wireless and embedded devices. Their methodology is the use of MPEG-7, which is a compact multimedia format suitable for small CPU devices. They desire to wrap multimedia content, as well as system preferences and other network properties within XML. The use of a binary XML representation enables their case set to reduce complexity, size, improve random access times, and the ability to use lossy representation of the data. They state that given their case set, GZip cannot produce the required results. Only with a binary XML representation can they reach the wireless and embedded devices. They also noted that using a binary format increases the robustness of their devices. This is an important property since they intend to operate in the wireless domain, known to be noisy and with high probability of lost data packets. For Siemens, XML is the ideal open source data format, and extending to a standardized binary representation is the next logical and required step for them to achieve their goals.

KDDI Research and Development Labs (2003), a low-bandwidth mobile devices optimization company, considers a binary XML as a means to extend the capabilities of cellular phones. They emphasized that XML-based applications are extremely desirable due to their interoperable format, but that mobile devices have limited capacity for complex and large files. Within the handheld domain, the parsing of XML is the most expensive operation so they desire a binary schema-aware XML format to marginalize parsing cost. Such a format delivers them a well-defined and pre-parsed format that is already in machine-readable form.

KDDI proposed a method called XML Document Encoding with Universal Sheet (XEUS), which is an encoder/decoder of arbitrary XML documents into a format conducive to cellular devices. A caveat to their proposed technique is it requires additional middleware because their XEUS relies on a gateway between the server and the devices for translation. To benchmark their XEUS, they ran it against traditional GZip and XMill

compression techniques. Their test cases were XMark documents, documents that are pre-formatted for compression, and Scaled Vector Graphic (SVG) documents.

While XEUS did not show significant compression differences when used against XMark documents, SVG did show significant improvements in compression as shown in Figure 11. They continue by noting that the decoding times on their cellular devices is about 14 times faster using the XEUS than the other XML encoding techniques, indicating efficiency improvements. A binary XML format on cellular and other mobile devices, such as XEUS, will extend their devices deeper into the Internet by allowing a richer content at higher rates for longer periods of use.

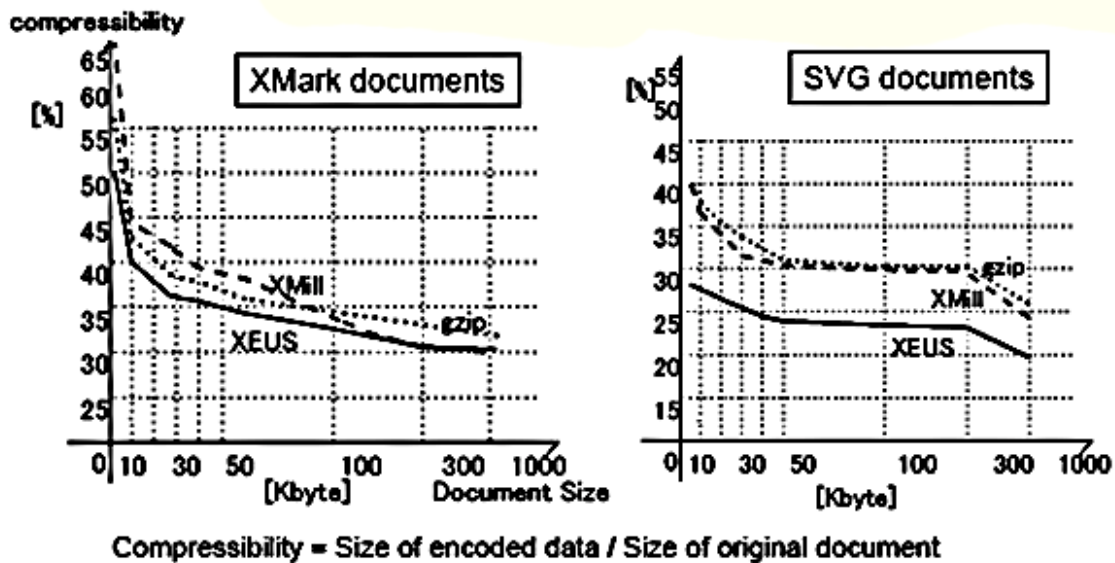


Figure 11. KDDI Research XMark Document Format Compression Comparison (From KDDI, 2003)

Hit Software Inc. (2003), a Database mapping and integration company, is using XML as their primary method of data exchange. They claim that as more and more messaging data shifts to an XML format, there is a risk of exceeding the capacity of communication channels. Based on that claim, they have conducted research to find ways to mitigate XML's verbosity that has led them to an event-driven compression of

XML that generates a message's data-structure while parsing the message stream. Their reasoning for this method is minimal memory footprint, and the relative ease of scalability of such a compression approach.

Nokalva (2003), a leading provider of standards-based software development toolkits and professional services in security and biometrics, uses Abstract Syntax Notation One (ASN.1) as an approach to binary XML. ASN.1 is roughly a universal schema that can describe information independent of system. ASN.1 can be used as an abstract definition of XML. ASN.1 is not XML, but is binary and does provide a standardized way to describe information. They found that while compression was not realized, gains in clarity and reductions in ambiguity from format specification are achieved.

Adobe Systems Inc. (2003) uses XML extensively throughout their products, and supports the W3C adoption of at most one alternative XML format. Their products are often embedded with multimedia and other data formats that are natively binary; going to a binary XML is a logical step, and an important requirement for them. They express the common concern that any binary format must support all domain-cases, and not focus on any one or common group of cases. Adobe recommended three potential alternatives for consideration:

1. Extending the XML Infoset to accommodate binary data – offers the advantage of not requiring a transformation on the binary data.
2. Define binary-aware, Post Schema Validation Infoset (PSVI) – this approach is problematic since it requires a valid schema associated with an XML document, which cannot be assumed in general. However, schema-aware XML compression almost always delivers significant compression improvements compared to all other techniques.
3. A packaging mechanism, such as multi-part MIME that permits binary data to be associated with an XML document – this approach fails to represent binary data within either the Infoset or PSVI, but does makes it applicable to XQuery.

Tarari (2003) is a developer of high-performance hardware-based solution for content inspection processors focused on network traffic and XML processing. It might be thought that a company that makes its money selling highly optimized XML hardware processors might oppose the development of a binary XML format, but they see it as an opportunity to merge their two main products through a binary XML. Given their business model outside of XML is already a binary format, and realizing XML is the dominant software input to their hardware; they believe they can realize increased gains by combining the two. They also see a standardized binary XML format as opening more opportunities to their technology than if various proprietary binary XML formats are developed. Tarari's single concern is that any standardized format must remain fully compatible with its hardware approaches to processing and accelerating XML.

BEA Systems (2003), an industry leader in communications equipment has expressed optimistic concern for any alternative XML binary format, especially that due process be taken to consider all tradeoffs of the numerous techniques, the performance gains, and requirements. Their expectation is that the existing XML requirements be maintained, and that all domain-cases remain intact. They conclude that if any alternatives are to be evaluated, then only one should be selected as a standardization solution.

Ontonet (2003) is in the biology ontology development market for the purpose of enabling easy machine processing of biomedical information and knowledge. Their focus is publishing on the Semantic Web. Their business model requires large sets of persistent XML data, normally processed using the DOM2 Core, which takes into memory the entire XML document. Ontonet is working on a binary exchange format to be followed by a compression technique. Their motivation is to enable larger volumes of persistent information resident in memory with faster processing.

L3 Communications (2003) has been developing communication systems since the 1970s. They have faced numerous situations where XML was the initial solution choice, but found the file size of XML under low-bandwidth conditions too limiting. L3 has set forth efforts to design a binary XML format to enable them to reach deeper into bandwidth limited networks with the XML language. They argue that XML's flexibility

warrant all efforts to find optimization techniques. Their efforts have been titled Common Message Format (CMF), which is a schema/DTD aware binary representation of XML. CMF was developed specifically for United States Air Force (USAF) Detachment 2 of the 645 Materials Squadron, and has been in successful use there since 2000. L3 has shown gains in processing and bandwidth utilization with their binary representation.

Systematic Software (2003) is in the business of structured information exchange for the healthcare and defense sectors. XML is the standard format for their information exchanges between customers who tend to operate in the low-bandwidth environments. Due to their operational environment, they have spent much effort developing a more compact XML format. With their years of DoD work, they have adopted a DoD-like messaging structure, Table 9, focusing on redundant tagging information supported by well defined data formats.

Systematic Message: <simple_time> <hour>08</hour> <minute>15</minute> </simple_time>
Legacy DoD equivalent: TIME/0815//

Table 9. Systematic’s XML Reformating Strucutre Example (From Systematic Software, 2003)

In their techniques, they keep only the actual information and essential tags. Given the well-structured formats of their data domain, this is all they need. Their proposed technique compresses XML based on the structure of the XML document using SSE/S, presumed to stand for Systematic Software Encoding/Structure, or with additional compression on the free text using Huffman coding SSE. To benchmark their results effectiveness, they conducted a comparison, Figure 12, against GZip, XMill, and legacy.

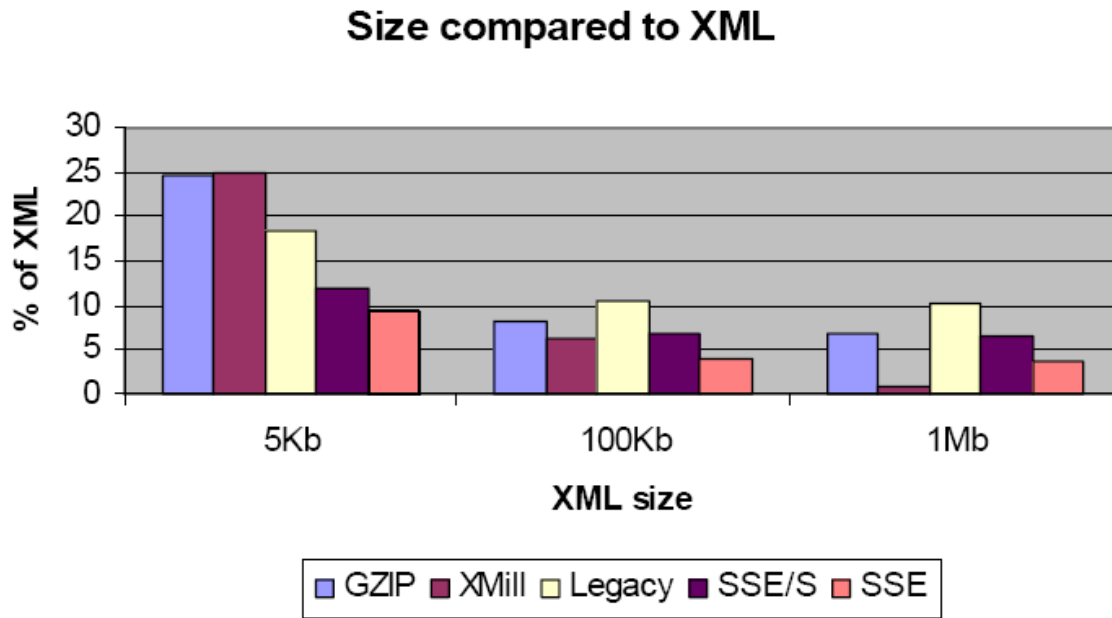


Figure 12. Systematic's Result Chart of Technique Comparisons (From Systematic Software, 2003)

The long run shows their structure-based XML compression does better than the other techniques. Given their operational environment, the low-bandwidth limit is their primary focus, i.e., compression performance is more important than document efficiency. Ultimately Systematic see a W3C-endorsed standard binary XML standard as beneficial to the Information Technology world in general.

Systems, Applications, Products in Data Processing (SAP) (2003), is a German company that develops integration tools that use XML as their data format for Web-based communications. Their domain of focus is point of sales (POS) transactions that are small files at high volume. They argue for the abandonment of the human-readable aspect of XML, stating that while XML was originally focused on ease-of-human use, XML today is used more as an exchange format processed by computers and not directly read by humans. Faced with a constant overhead associated with each XML-based transaction per communication channel, regardless of document size, their high-volume problem domain suffer limiting bounds on the maximum number of transactions possible per time period using the native XML 1.x format. While they have not developed a custom solution, a more efficient interoperable format will enable their domain to grow.

Their conclusions are that any alternative format must be easy to implement, be Web supportive such as fragments, fit into the Web stack, and most importantly, it must meet real-world usage requirements.

Media Fusion Corporation provides XML-based storage solution, and boasts they are the first to focus on technologies that use XML (Media Fusion, 2003 & n.d.). They are working to deploy XML into the cellular phone and small appliance sectors. Their problem is the amount of processing and memory usage required for XML is impractical for their small devices. These devices are limited due to size, CPU power and memory capacity. They have focused extensive efforts to make a compact and pre-parsed binary XML format to eliminate a majority of XML's limits on their small devices. Their concept is a DOM-like format that consists of two parts: the structure of the document and the unique strings of the document. These parts are formatted in byte streams that are consistent with DOM serialization, then merged at the receiving unit. Because both parts are in binary format with redundant data eliminated, a compact and efficient format is achieved. Essentially, this approach sends the data with indexes to the location within the document where the data belongs. Their method has not been put into mainstream application, but has demonstrated the ability to push XML onto small devices, including those that can be found on appliances.

Canon Information Systems Research (2003), a part of the Canon printer company, is tasked with research and development of digital imagery technologies, specifically the 2D XML Scalable Vector Graphic (SVG). Within their domain of study, they found that the traditional bandwidth argument of XML is a secondary issue, that the processing of an XML document within a base64 coding is the primary issue. Bandwidth, they argue, can be addressed by throwing more bandwidth at the problem, by buying more access. It is the XML encoding and decoding cost that remains persistent regardless of bandwidth. They conclude that the W3C ought to reconsider the text only design of XML, and review new XML formats that can maximize bandwidth and processing efficiency.

Sun Microsystems (2003) is heavily invested in Web servers and Web services. Sun's current XML solution within the Web services domain is to leverage the network

architectures so XML alteration is not needed. Their Web services approach wraps the data or acts on behalf of requesting users, and does so in well-defined and optimal methods. They note that outright compression of XML will help with bandwidth, but that the added complexity and time that the sender and receiver spend during compressing and decompressing a document might not warrant the effort. Sun suggested that working each layer of the standard Web services stack, as shown in Figure 13, may deliver better performance.

1. The transport layer is generally HTTP.
2. The infoet layer is an XML API for dissecting the data from within the document.
3. The binding layer datatypes the XML values to the defined Web Services datatypes or otherwise considered schema data typing.

Standard XML Pipeline

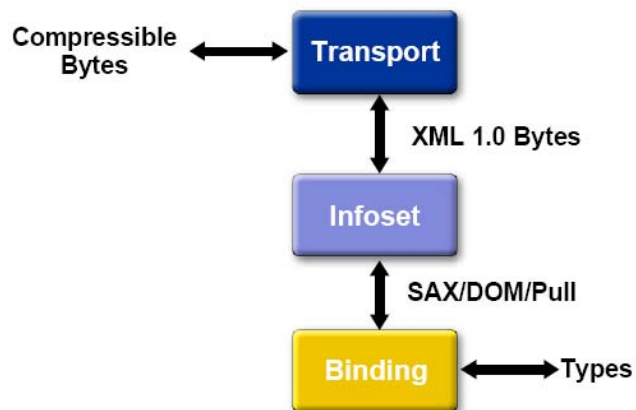


Figure 13. SUN's Traditional Web Services Pipeline (From Sun, 2003)

Sun recommends two alternative solutions to combat XML's problems:

1. Replace XML 1.x with an alternative binary format that is self-describing and lossless, but states this achieves inconsistent results as the document size changes because the binding operation must still be performed. Figure 14 provides a rudimentary example of this concept.
2. Eliminate the info set layer, and convert the XML to a schema-informed format that is easily processed by the binding layer, as shown in Figure 15.

XML and Fast Schema Encoding

XML	Fast Schema
25B <stringT>string</stringT>	7B string
29B <integerT>12345678</integerT>	4B 12345678
25B <booleanT>true</booleanT>	1b true

Figure 14. SUN's Recommended Self-Documenting, Length Prefixed XML Encoding (From Sun, 2003)

Fast Schema Pipeline

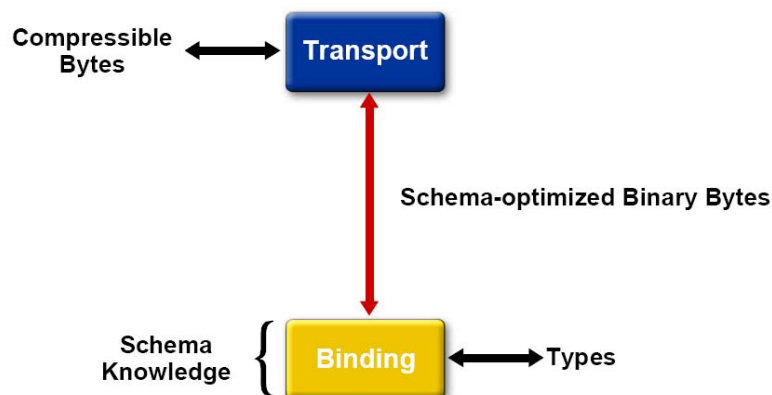


Figure 15. SUN's Recommended Fast Schema Pipeline (From Sun, 2003)

Sun's experiments shows that the Web service domain might achieve large gains from a binary XML format by passing some of the complex processing to different devices within the pipeline, and if done so with a schema, as listed in Figure 16 and Table 10, can deliver speeds of up to 10-times faster processing.

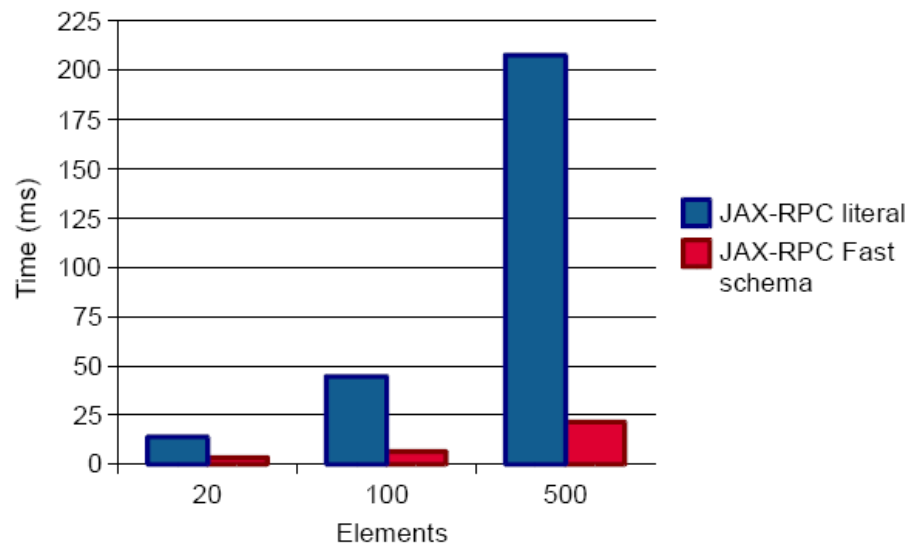


Figure 16. SUN's Comparison of Results Chart between Traditional XML and Fast Schema (From Sun, 2003)

Factor Measurement	XML	Fast Infoset	Fast Schema
Size	Large	Medium	Small
Processing	High	Medium	Low
Self-Describing	Yes	Yes	No
XML API support	Yes	Yes	Yes

Table 10. SUN's Table Result Property Comparison (From Sun, 2003)

Nokia (2003) uses XML as their data exchange format for a wide range of uses, primarily focusing on Web services, such as in Web-browsing, multimedia and messaging. Nokia states they do not believe any one binary XML format can capture all domain-case needs, but they remain engaged in the study. Even though they make their money on proprietary developments, they believe any alternative XML format must be an

open standard, and that it should not require any specific domain awareness to operate, or as they put it, “wireless aware but not wireless specific.”

The W3C Timed Text (2003) working group is concerned with audio and visual file representation for real-time applications. They are deeply interested in a binary XML format given their data domain is natively binary, and XML is often their transport vehicle. Their domain-case has some unique needs outside of the common bandwidth minimization and processor complexity:

- The ability for playback from arbitrary points within the file: Random Access.
- The ability to transmit partial subsets: Fragments.
- The ability to dynamically update previously transmitted data without retransmitting unaffected data: State Persistence.

While they have not developed a solution of their own, their primary focus of comment is to ensure their domain-case receive adequate consideration and support for an alternative XML standardization consideration.

CubeWerx (2003) is focused on openGIS (Geographic Information Systems), which is derived from the Geographic Markup Language (GML), both of which are members of XML family of languages. These XML languages are extremely dense in numerical data, having little more than the XML element tags as the only repetitive text found within their documents. This lack of repetitive text values (since typically the contained numeric values do not exactly repeat) makes GML and openGIS completely inapt for the common compression techniques. CubeWerx desire a binary format for XML to take advantage of faster numeric processing and smaller file size, but expressed concerns that any solution that truly meets their needs will likely not translate well to other XML case domains; binary normally does not equate to interoperability, and generally focuses on a particular domain. They also wrote that while making the smallest document possible is a good idea in many cases, it is not necessarily a monotonic good since a smaller document could make local representation more complicated.

Although a binary format is not easily made domain unspecific, and does abandon the human readability, CubeWerx claims specifically that the human readability can easily be maintained. Their logic is few people edit XML files directly outside of an editing tool. Their suggestion to the human readability debate is to let the editing tool do the translation, binary to text to binary. Any alternative binary XML format will have to be encoded and decoded by an editor regardless. Let the editing tool decode the binary XML to ASCII for presentation at the editor, and then save the human edited XML to the alternative binary format behind the scene the same as any word processor does. For most intended purposes, this method preserves the human readability of XML with a binary XML format. Of note however is that partially garbled compressed documents might not be easy to convert or correct. CubeWerx's recommended solution to XML's verbosity and performance for numeric-based documents is to focus on a narrow set of datatypes, and indexing to the redundant values. By limiting the number of numeric datatypes (int, float, array of float) a binary format is more compact.

Using their binary format, Binary XML (BXML), a traditional XML document can be reduced to almost half its original size. Table 11 shows some examples of the gains they achieved on XML cases that consisted of primarily numeric data. CubeWerx does note that these compression gains are on strings of UTF-8, that UTF-16 or other string encodings might not deliver similar results.

File	XML	BXML	XML + GZIP	BXML + GZIP	XML + BZIP2	BXML + BZIP2
WMS Capabilities	935	521	79.1	76.6	53.6	56.4
OpenOffice.org Doc	702	427	76.6	75.5	52.1	57.6

Table 11. CubeWerx Comparison (in Kilo Bytes) of Proposed BXML and XML
(From CubeWerx, 2003)

Because their binary XML format eliminates most of the redundant information, the continued gains from zip or other compression techniques diminishes quickly, that is, the gains of applying a zip technique to their BXML is not significant to zipping the raw XML document. However, by not compressing the document, the receiving station does

not have to perform decompression. Additionally, since their BXML file format is nearly all binary, its raw format performance is high compared to other data formats.

Advanced Technologies Group (2003) makes interactive satellite distributed video content, called Programmes, for TVs, set-top boxes (STBs), Digital Video Recorders (DVRs), PCs, mobile phones, portable media players (PMPs), removable media, and other small devices. Their format for content dissemination is XML. Their satellite networks do not have an always-on bidirectional communications capability so they have to communicate in a continuous loop over different channels of varying bandwidth capabilities. Figure 17 shows their general content delivery flow.

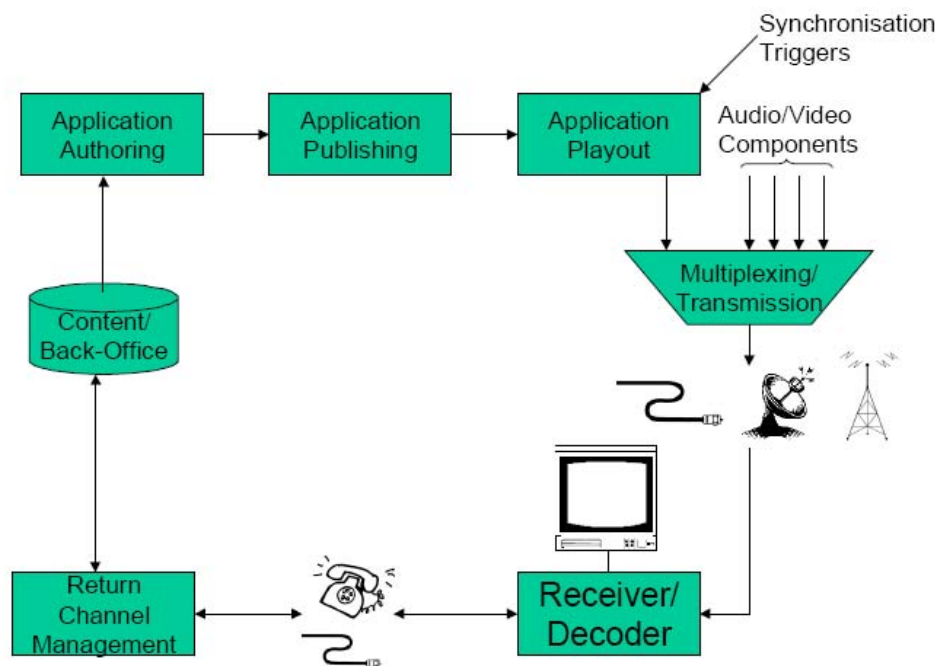


Figure 17. Advanced Technologies Group, NDS Satellite Communication Loop Diagram (From Advanced Technologies Group, 2003)

Advanced Technologies Group's binary XML formats transform XML to binary at the application play-out module, and is incrementally updated on-demand, normally through a telephone from customers. The real problems arise at the Receiver/Decoder due to its limited processing and memory abilities. The updates being sent to the Receiver/Decoder need to be as small as possible, and the data needs to be structures so that the devices can work off data fragments. Because most updates are larger than the

Receiver/Decoder can maintain, Advanced Technologies Group is investigating alternative XML formats to achieve greater compression of XML. Table 12 lists some averages in terms of file sizes using their techniques.

XML Format	8 PROGRAMMES	4 PROGRAMMES	1 PROGRAMME
Source XML	9317 bytes	4868 bytes	1350 bytes
DVB SI EIT	1050 bytes	531 bytes	154 bytes
MPEG-7 BiM	991 bytes	497 bytes	148 bytes
ASN.1 PER	1049 bytes	525 bytes	153 bytes

Table 12. Advanced Technologies Group, NDS Table of Sampled Alternative XML Format Techniques (From Advanced Technologies Group, 2003)

Advanced Technologies Group concludes that drastic file size savings can be achieved, and that without these savings, “it would be uneconomical to broadcast many of the interactive applications that are deployed today.” They also add that the reduced file size and prepared format is what enables the existing receivers to operate on complex events.

As the Internet continues to grow, Web services and Semantic Web technologies are increasing as the vehicle of choice for data exchange. TeliaSoner (2003), studies these technologies, nomadic services as they put it, for the intended purpose of pushing them to mobile devices. What they found is the verboseness of XML has made the wide-scale use of XML unrealistic on mobile devices due to bandwidth limitations. For mobile devices, the bottleneck is the wireless network, and not the processing and parsing of the XML document. They suggest that beyond a reformatting of XML to binary, that the network protocols be reviewed. Optimization to the network protocols might achieve the performance gains all domains need without altering the XML format itself.

The Computer Science Department at the University of Helsinki (2003) studied XML and wireless networks. Like many of the previous arguments, they argue that XML is excellent as long as it is not used within low-bandwidth wireless networks. In an effort to address these wireless limits, they developed a general-purpose event-driven binary

XML format to reduce the size and parsing complexity of XML documents. Their approach is to store caches of XML events, such as SAX events, about the documents in memory, and uses these as pointers. As the cache repositories are built up, the binary XML document becomes a list of indexes into cache, which are represented in fewer bits than the raw string character bytes. Beyond this event-driven structure, they also present optimization techniques such as pre-fetching, element caching and item omission:

- Pre-fetching is simply filling the caches with known common data before runtime.
- Element caching adds a new tag for the elements where the element name is in cache with its associated index.
- Item omission is nothing more than leaving out unimportant information from the XML document, or leaving out items that are standard in every document.

The product of their method is a binary file of indexes to caches that contain the original XML document's string content, eliminating all redundant information and represents the XML document in fewer bytes than native XML.

Sosnoski Software Solutions (2003) has developed a similar binary XML format to that of the University of Helsinki, but instead of byte-sized indexes, they use dynamic growing bit-sized indexes. The number of bits reserved for the indexes into caches of information is dependent upon the number of entries in cache. For example, if only 60 strings are present in a cache, then only 6-bits are needed to adequately index to each string in the cache. If greater than 128 strings are in a cache, then this index range grows by two additional bits to accommodate the increased size of the cache. Sosnoski's motivating intent is to reduce processing overhead by designing a binary document model. They structure SAX2 events to build up a transformed binary file that they call XBIS, presumed to stand for XML Binary InfoSet. Their test cases consists of three XML files: periodic.xml (periodical table of elements), SOAP2.xml (a SOAP message file), and xml.xml. Their test files are then applied to XBIS as well as Piccolo and Xerces, which are competing binary XML formats. Their goal is increased efficiency,

and as depicted in Figure 18 and Figure 19, their method delivers the fastest binary XML encode and decode times among there choices.

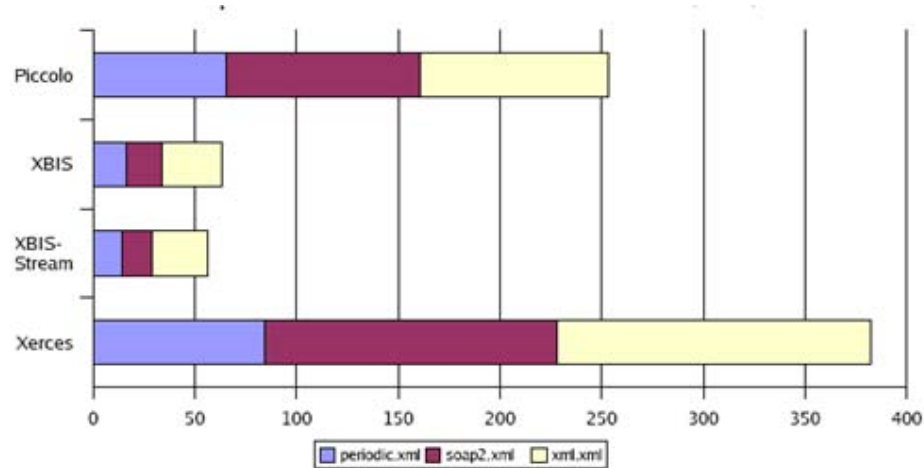


Figure 18. Sosnoski Encode Time Comparison of XBIS and XML (From Sosnoski, 2003)

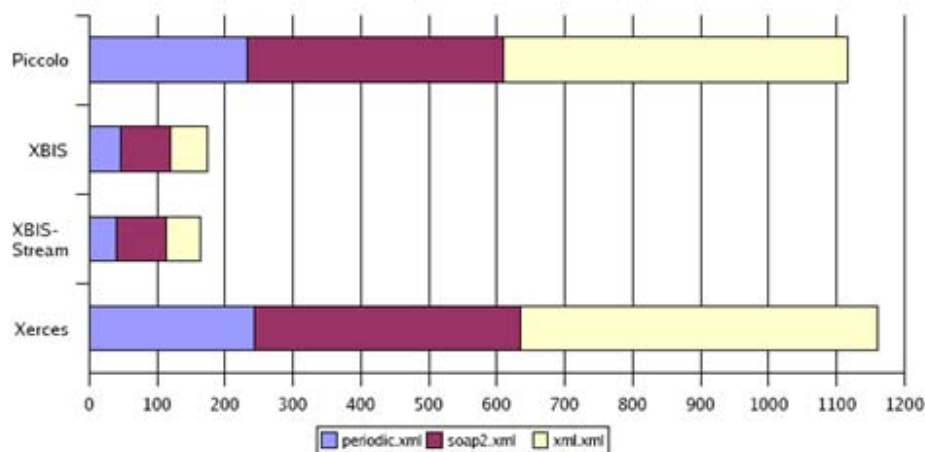


Figure 19. Sosnoski Decode Time Comparison of XBIS and XML (From Sosnoski, 2003)

Sosnoski’s overall representation produced a much smaller and faster file format. Their algorithm is complex, but it appears to work on any arbitrary XML document. The only note of potential discrepancy was their removal of DTD information in all test cases, which would exclude DTD-focused XML from optimization under their technique.

Stephen Williams (2003) is an independent software developer involved in XML consulting. His experiences have come from the financial sector, engineering fields, and traditional information technologies. His XML considerations are based on his independent and unbiased experience from a wide gamut of industry experience (Williams, 2009). He addressed most of the same arguments for an alternative binary XML format already listed: file size reduction, efficient binary format, not tied to a particular schema, and support for non-XML data insertion such as images and video. Being independent of any controlling corporation, his unbiased desires for an improved XML format provide a further confirmation of the benefits of an efficient binary XML format.

Lionet (2003) has been using XML as the core of its software developments since 1997. They argue that a more efficient XML format is essential given that they, like most others in the XML business, have pushed the boundaries of XML to the maximum. Lionet's method to transform traditional XML documents into a binary format is similar to the indexed approaches previously listed. They create a number of tables: Namespace, Attribute name, Attribute value, Attribute, Tag, and Text. Each table row is used as an index to efficiently represent the original XML's textual data, but within a binary format. Based on the data location within the XML document, instructions, which are SAX events, are created with parameters based on the indexes of the tables. The result is that all text occurrences within the XML document are encoded as text one time only, and all follow-on occurrences of the same text are represented by an index. Using their binary XML algorithm, they claim a number of significant reductions in processing intensity and memory usage, as well as an increase in efficiency. They list parse-time reduction by a factor of 2 to 10, file-size reduction by a factor of 6 in bandwidth requirements, and a memory-footprint reduction factor of 3 to 4.

The Consultative Committee for Space Data Systems (CCSDS) Working Group is focused on developing solutions for end-to-end data exchange within the space industry (CSC/NASA, 2003). The collection of information from space operations is difficult and

expensive, and as depicted in Figure 20, comes from many sources. CCSDS leverages the economy of scale, waiting until huge amounts of information have been captured before transmission.

CCSDS is reviewing the best methods to systematically store and discriminate the captured information, with XML being an initial option of review for a number of space considerations.

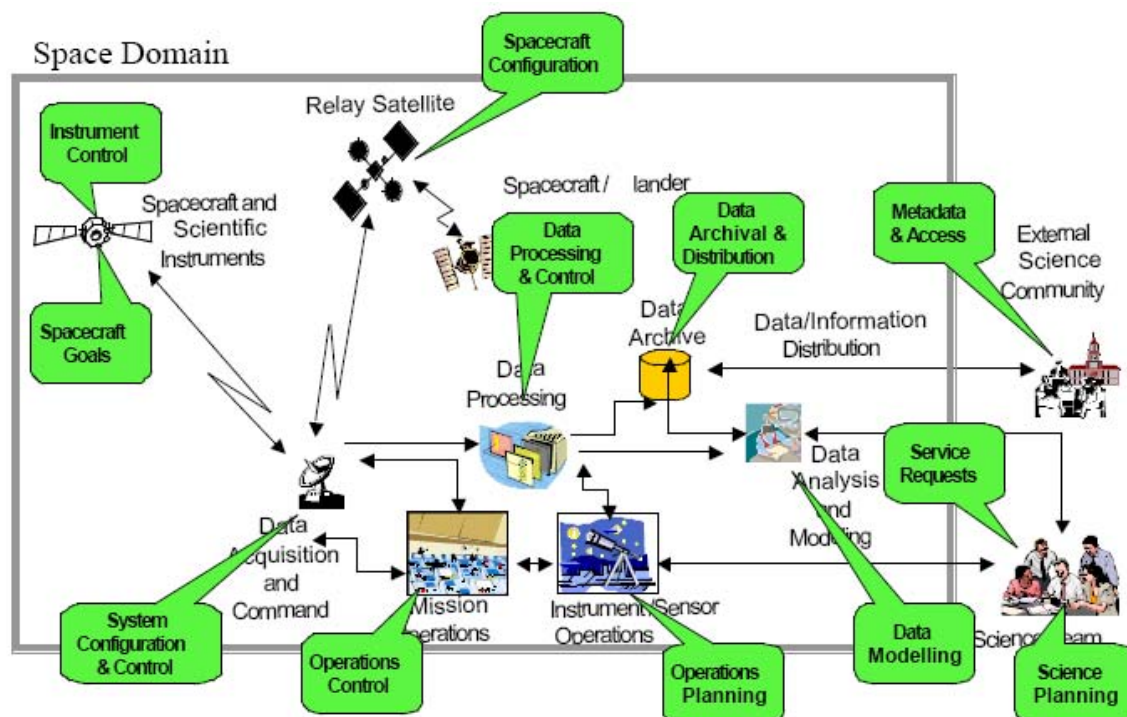


Figure 20. CCSDS Space Domain Functional Areas for XML Implementation Considerations (From CSC/NASA, 2003)

In their consideration of XML, the verbosity is of primary concern without consideration of efficiency. Their primary concerns, and support of XML for the space domain:

- Why XML
 - Vender Neutral
 - Self-documenting, removes scientific-format ambiguity

- Why Not XML
 - Verboseness of XML and the inherently large files of space
 - Space use cases data is primarily numeric {arrays, tables}

CCSDS points out that using XML creates a file of 2 to 4 times the size of their current file format, and that the tree format of XML is a poor structure for some space data. CCSDS has since focused on the use of XML as a data archives that can store their data efficiently and compactly using the common zip tools. They continue to research additional encodings to support base64 data, and have not abandoned XML.

Expway (2003) is a provider of content for both television services and mobile devices, called Electronic Service Guides (ESG). Their domain is interactive, filled with advertisements and data-casting, all of which is on-demand with end-user orientated formatting. Their focus of study has been the delivery of their services via an XML format, moreover, an efficient XML format. Optimistically they believe the argument that no one binary XML can meet the needs of every domain-case can be overcome. Their competitors often use the “Doctor it hurts (XML is too bloated) when I do this...don’t do that” comedy line as their excuse to abandon XML. However, Expway see the benefits of XML in its quality tools, easy integration, and easy development as reason enough to work on overcoming the file size problems of XML documents. They also boldly state that when someone asks for smaller XML, often what they are really desiring is just faster XML.

While a smaller file size will reduce transmission time, it’s then the processing of the XML that absorbs the most time consideration. Processing XML is expensive due to format, type conversions, such as text to number, embedded binary files, and querying for random access. For Expway’s domain-cases, random file access is the most important consideration to allow its customers the ability to view what they request, and change the layout rapidly.

Expway has explored a number of encodings as possible alternative XML-formats with varying degrees of success, routinely finding that what seems to be perfect for one-domain ends up being weaker for another. They nevertheless believe they have tested a

sufficiently large sample of domains to warrant a movement for standardization. They remain optimistically hopeful a single optimized format will be adopted. Expway concludes with a strong argument for standardization and they term the plethora of incompatible alternatives as Balkanization. The time has come to develop and implement a binary standard before the spread of XML deepens, making the transition problem more complicated.

Agile Delta (2003) has long worked with the DoD in creating an efficient XML encoding that will enable the Network-Centric vision of a systems-of-systems for knowledge and information sharing. Agile Delta has been in the field of study of efficient XML representation since 1995 with their first design called the Knowledge Based Compression (KBC) based on the principles of information theory by Claude Shannon. Their general concept is to represent the XML information with the fewest number of bits needed to uniquely identify the data, that is, order $\log_2(n)$ bits are needed to represent n different pieces of information. However, if some of the n pieces of information are more likely than others, even fewer bits could be used if a non-uniform index bit length is considered. This results in order $\log_2(1/p_i)$ bits to represent data that has a probability p_i of occurring. They clarify this with the classic “one if by land, two if by sea” scenario from the American Revolutionary War to convey two important messages with a single signal. That is, one light to indicate that the British were coming and second light to indicate how they were approaching.

Using Agile Delta’s reduced bit-representation approach, the file size of XML drastically decreases, reduces bandwidth load, extends battery life on mobile devices, and increases performance. Together these benefits provide the potential to increase the market for XML by allowing it to reach CPUs that previously could not process native XML. Their solution addresses most of the concerns in the market and those of the W3C XBC working group, and also illustrates that some of the W3C XBC desired features are not realistic in a single design, and should be implemented via a second layer. Examples of such features are random access to the XML content, dynamic updates, and the mixing of text and binary. They conclude similar with Expway, that the time has come to enact a standard before the already complicated problem worsens.

2. Arguments against a Binary XML Format

The Microsoft Corporation (2003) was against the development of a standardized binary format for XML. They claim to point 3 of the *10-points of XML* document that XML is text. Text helps people learn, text is simple and there are many tools to optimize text parsing. Additionally, native XML is understood and is used as is within the IT industry. They state a new standard would add complexity to the XML space, requiring vendors to support two distinct versions of XML instead of one, and both vendors and users have suffered the growing pains of the ever-increasing families of XML languages. They conclude the contrary of the two predominant arguments for a new standard:

1. XML verbose: They claim that if compression is the goal of a binary XML format, then GZip or XMill can achieve compression good enough for low-bandwidth. Both of these compression techniques are designed for text formats and do show significant compression of XML files.
2. Processing intensity: Stating that if the entire XML document is in binary, then everything would have to be parsed and processed, adding complexity and defeating the benefits trying to be accomplished for small handheld, “it becomes a tradeoff between smaller memory footprint and higher parsing cost, which consumes more power.”

Microsoft also notes that a large number of XML 1.0 applications are already deployed and changing this installed base is extremely costly and difficult.

The Oracle Corporation (2003), a leader in database technologies, argues that the routine nonproprietary method of XML processing is the Document Object Model (DOM) which processes an entire document into memory. Bringing an entire document into memory can quickly run an application out of memory spaces as the XML document grows in size. The other method is to use Simple API for XML (SAX), which is an event-driven method that saves on memory space, but it has to rely on sequential XML processing and proprietary methods. Oracle suggest a need for a XML compression to enable DOM-like XML processing to ensure platform independence for working in the Internet or direct-communication domains. Oracle contends that the purpose of the

compression is dependent upon where in the processing chain the document is operating. The requirements for mobile device processing and displaying of XML are different from a Web service application or that of an enterprise server. They remain skeptical as to whether or not a single case of compression can meet the unique needs of all domains, and did not yet support an alternative XML format.

Computer Engineering and Networks Laboratory (2003), is a Swiss company focused on computer communications and distributed networks. While Computer Engineering and Networks Laboratory remains interested in a “good idea” and the efforts of a binary XML format, they believe this needs to be a starting point rather than an end state. Their recommendations are to reconsider the foundations of XML and to define a binary representation based on existing well-known algorithms rather than a new compression technique.

XimpleWare’s (2003) business model is the delivery of SOA tools for enterprises around the world. They are not a firm believer that a binary XML format will help the XML information set, claiming that the often-mentioned verbosity and performance issues of XML are seldom the true limiting factors. They do acknowledge the verbosity of XML in low-bandwidth environments is a concern, and processing is of concern regardless of bandwidth. They believe that XML is its own problem claiming that processes running on XML when compared to non-XML always underperform. They acknowledge that while an alternative format is likely to deliver some improvements, doing such is contrary to XML “...give up the luxury of reading the wire format...back to the dark ages,” that is, reverting back to a legacy style of data formatting before human readability became widespread. Reverting to the “dark ages” in terms of XML would mandate a persistent schema, the same type of rigid format problem that originally enabled XML’s loose-format success over file-structures. Their solution is not the format of the XML, but the processing of XML by keeping the entire document in memory, much like a database to enable exceptionally fast queries and restructuring. This approach keeps the schema-free features of XML, as well as the self-documenting human readable format. The essential arguments are that memory is cheap so buy more memory, invest in faster XML parsing techniques, and don’t jeopardize XML’s successes.

IBM (2003), a maker of all things digital has worked with XML as their dominant exchange data format for years, and has made a number of IBM-specific attempts to improve the XML verbosity and performance. IBM points to four success of XML:

1. Only one standard XML format with limited character settings (UTF-8 or 16), but can encompass any number of specific characters with escape features.
2. XML is human readable text, not binary...industry is focusing on more text than binary.
3. Flexible format that can reach to nearly any domain.
4. XML is self-describing.

The successes of XML are also the roots of XML's problems. IBM, however, claims that any alteration to these keys of XML's success will undermine XML's interoperability since "...binary XML proposals with a healthy reluctance to tinker with the formula that has successfully carried XML so far." They do believe that an alternative XML format would make the XML family better, but also believe the diversity of opinions and needs will not converge. Moving forward without standardization and convergence will lead to failure. IBM concludes that each domain might have to develop its own binary XML format in order to achieve domain-specific goals, but that this inhibits inter-domain operability, which would be an overall failure.

Rick Marshall (2003), an independent software developer with 25 years of experience within database domains and XML representation, claims that databases are the most efficient format, and that a binary XML is not worth the effort or the troubles of decoding. He states that while XML is a human readable format, once namespaces are introduced into an XML document the readability is questionable even though it is text. Rick does note that XML tags could use improvement, as they are part of the reason for XML bloating. In addition, he notes for many documents, finding the end tag is complicated, and so adding a tag-detection mechanism would be an added benefit. He further claims that compression is highly dependent on the use-case domain, and finding an efficient universal compression "silver bullet" applicable to all domains is not likely to

ever be achieved. His final claim is that Moore's law will "take care" of the processing problem; optimization of the hardware should be addressed as the real long-term solution to XML's problems, not the XML format itself.

Software AG (2003) is a developer of database management systems with customers around the world, but with primary focus in Europe. Like the other database companies, they see the problems of XML not as a function of verbosity but of efficiency in processing XML. They present external XML solutions:

- The Moore's law argument stating "...processors are under-utilized and Moore's Law predicts processing speed and memory capacity will double every couple of years...", the XML problem will not be a problem in the near future given Moore's law.
- Address the XML problems with a hardware-specific solution similar to a graphics card, and not the redesign of XML. The justification is that if XML is going to be prevalent, special optimized hardware is justified and has been developed by a few companies that can process XML faster than a generalized CPU.
- Optimize XML's tags for faster querying, improving processing efficiency.

Software AG's analysis doubted that any one binary format can meet the processing needs of both well-formatted schema and schema-less XML while at the same time addressing the verbosity issue.

E. SYNOPSIS OF INTEREST FOR AND AGAINST AN ALTERNATIVE XML FORMAT

A quick summary of the key points for and against an alternative binary XML format:

1. Summary of Arguments Pro Binary XML Format

- Because of the high frequency of XML document processing, a XML-specific compression technique is justified; the economy of scale justifies the upfront cost and development burden. The cost of added software development is offset by the end benefits of smaller and faster documents able to reach uses cases previously unable to utilize XML.
- Streaming data in XML format will directly benefit from a binary representation. Working in a binary-only format for most multimedia will produce a savings of time and processing since multimedia data is natively binary.
- Numerical heavy documents share the same argument as multimedia. Engineering and simulation tools generate XML with more numeric data than text, so common text-redundancy based compression tools cannot deliver file size savings. This is also true for Web services and other data-sharing methods. Retaining the natural binary format of numeric values delivers efficiency and compactness instead of placing numbers into strings knowing that GZip and the other text compression tools will not work effectively.
- Pure binary formats allow XML documents to leverage the many error-correction techniques that have proven useful, but are not applicable to text. Using these correction techniques, XML can be extended to poor quality transmission lines with an assurance of service.

2. Summary of Arguments Con Binary XML Format

- A new compression technique will result in the loss of the human readability of XML, a cornerstone of XML's success.
- If XML is processed in such high frequency, perhaps a XML-specific devices that is optimized for XML processing is the answer, similar to the specialized functions of a video card.
- Given the Moore's law that processing power doubles every few years, let the processor or hardware do more and the XML problem will just go away with time.
- Another consistent concern of a binary XML is its reliance on a schema. A reliance on a schema will deny many XML documents from the alternative format because they do not have a supporting schema.
- It is questionable whether the complexity of a new format is worth the effort given GZip and other text-based compression tools already in the market can produce significantly smaller documents than those of the original. These opponents do recognize a XML-specific compression technique produces better compression, but in many cases, the compression gained compared to GZip is minimal, and is computationally expensive to perform.
- A single alternative format is not uniformly good for all domain-cases. For the cases where a binary representation provides significant benefits beyond that of GZip, it is argued that relevant domains should develop their own custom compression, but not a standard that all XML implementations are required to adopt and implement. Once a binary standard is established, all XML implementation will need to implement two versions of XML.

F. CHAPTER CONCLUSION

The primary rationale for a compressed binary XML representation is to grow the Web by enabling XML to serve use cases that otherwise are unable to support the native text-based XML 1.x language. Both DoD and the business world have a vested interest in an alternative binary format for XML. There are cases of exception to a binary XML, but even those exceptions do acknowledge the theoretical benefit of a standardized compressed binary XML format, just not the format specification itself. Many believe the need exist, but doubt a single binary XML format can meet all the requirements of every domain. From the discussions of the binary XML case reviews, what works in one domain will not carry in full to every other domain without either extending the XML design or relaxing the W3C XBC working group requirements.

G. CHAPTER SUMMARY

This chapter discusses the historical awareness of XML's verbosity and processing problems. The W3C's XML Binary Characterization working group (XBC) efforts to resolve XML's issues are summarized from the perspective of both DoD and the business world, noting the pros and cons of an alternative XML format.

VI. W3C BINARY XML FORMAT (EXI) DECISION JUSTIFICATION AND FRAMEWORK

A. INTRODUCTION

This chapter defines the evaluation methods used by the W3C to test candidate compact binary XML formats. The verification and validation of nine different techniques are presented, with EXI being the technique with the best overall scores that received W3C endorsement for continuation in the standardization process. EXI is further compared with GZip to provide baseline comparisons against to the current industry standard compression technique. This chapter concludes with the W3C general recommendations for EXI implementation and usage impact statement.

B. VERIFICATION OF CANDIDATE BINARY XML FORMATS: TESTING FRAMEWORK

The W3C considered nine candidate techniques in pursuit of the best alternative binary XML format. Each of the candidates was validated against a collection of XML test cases from the W3C XML Test-corpus to verify that each of the candidates operated as claimed. This comparison was required by the requirements published in the *Efficient XML Interchange Measurements Note* (W3C, 2007). The generalized flow of candidate testing is based on Figure 21, with statistical checkpoints retained during the process to measure the compactness, efficiency and round-trip accuracy of each candidate. The ultimate goal of the testing was to find the best candidate technique and recommend it for standards development by the W3C.

Based on the results of the candidates testing, EXI became the alternative XML format technique pursued by the W3C for standardization.

Notional EXI Test Corpus & Measurement Overview

Motivation: define consistent EXI terminology for diverse document sets and measurement algorithms

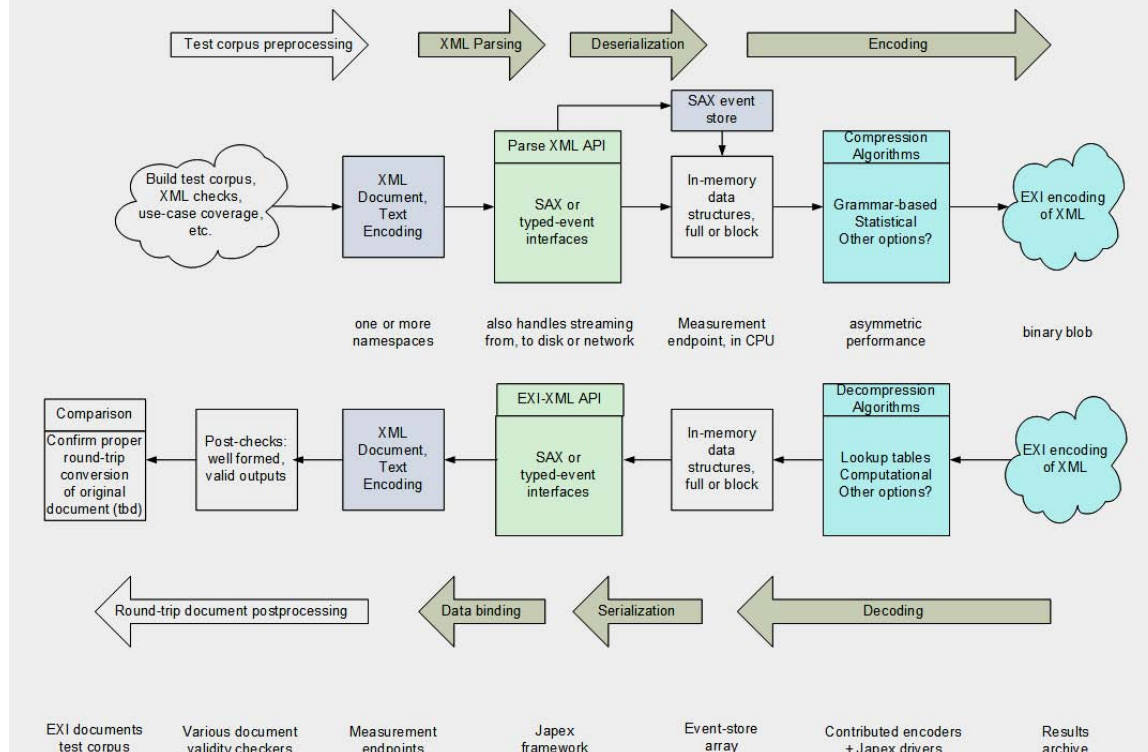


Figure 21. W3C Testing Framework Flowchart for Candidate Binary XML Formats (From W3C, 2006)

1. Testing Framework - Measurements

Using the original work in XML Binary Characterization (XBC) working group, both file size and the complexity of XML are used as measures of effectiveness to evaluate the candidate techniques. The natural choice of measurement for size is byte-to-byte comparison of the original file to the compressed file, and for the complexity the total time to encode and decode a document. A binary XML candidate has to achieve superior results in both the decoding and encoding with the decoded document being XML “syntactically equal” to the original document. This concept of syntactically equal is subtle, but important to understand. A poor algorithm might be superior in one direction, such as encoding, or deliver a smaller file size, but fail to deliver the original

document at decode; an unsatisfactory algorithm. Any alternative technique to continue for standardization has to be faster and smaller than existing practices, and must always be accurate, otherwise continued effort is not justified.

2. Testing Framework - Test-corpus

To test and measure the candidate techniques, the W3C developed a test-corpus of XML documents from a collection of more than 10,000 submissions that span the range of the XML family of languages and vocabularies. The test-corpus yielded eight categories of documents, called the Use Groups by the W3C, as listed in Table 13. Figure 22 lists the test-corpus documents in terms of their content density, i.e., sum of “values” divided by total file size, organized by the family with which the test case is associated. Content density is a metric indicating the complexity of the document, that is, how much of the document is information relative to XML structure.

The purpose of the test-corpus is to avoid selecting formats that achieve good results at the expense of being narrowly focused, biased towards one domain over another, and to help determine the generality of each candidate.

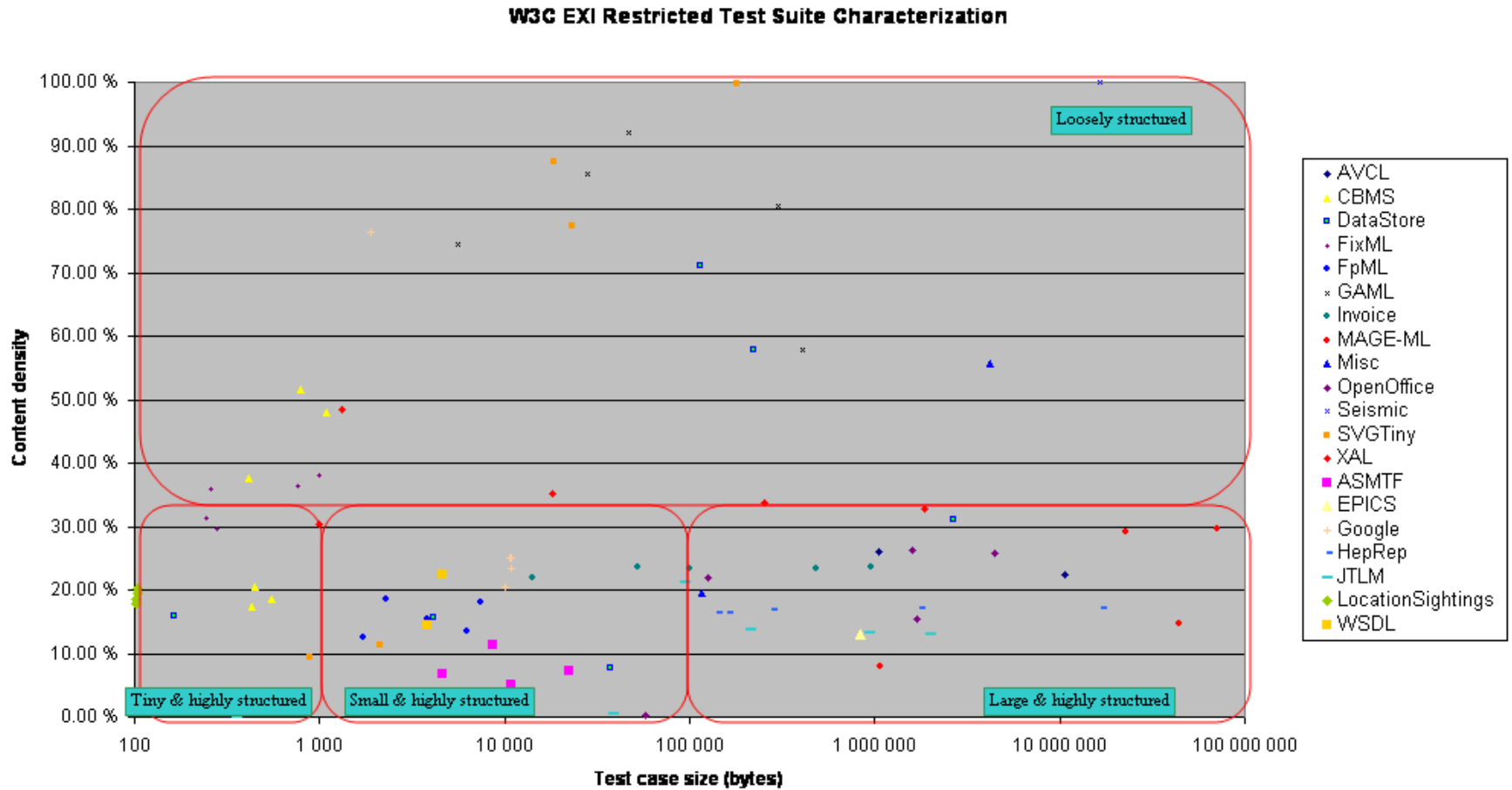


Figure 22. W3C Binary XML Test-Corpus of Documents by File Size and Value Content Density (From W3C, 2007)

Use Group	Description	Cases Included
Scientific Information	This covers data that is largely numeric in nature, used in scientific applications.	GAML, HepRep MAGE-ML, and XAL
Financial Information	This use group includes cases in which the information is largely structured around typical financial exchanges: invoices, derivatives, etc.	FixML, FpML, and Invoice
Electronic Documents	These are documents that are intended for human consumption, and can capture text structure, style, and graphics.	OpenOffice, SVGTiny, and Factbook
Web Services	This use group consists of documents related to Web services, both messages and other types of documents.	Google and WSDL
Military Information	These documents are encountered in military use cases.	AVCL, ASMTF and JTLM
Broadcast Metadata	The type of information in this use group captures information typically used in broadcast scenarios to provide metadata about programs and services (e.g., title, synopsis, start time, duration, etc.).	CBMS
Data Storage	This use group covers data-oriented XML documents of the kind that appear when XML is used to store the type of information that is often found in RDBMS.	DataStore and Periodic
Sensor Information	Documents in this use group are information potentially provided by a variety of sensors.	Seismic, epicsArchiver, LocationSightings

Table 13. W3C Binary XML Test-Corpus of XML Documents Listed by Use Group
(From W3C, 2007)

Because any efficient binary XML format must support schema-informed as well as schemaless XML documents, Test Application classes are designed to exercise combinations of schema-informed (or other document descriptor) and schemaless. Table 14 list the W3C application classes along with the descriptions used in the framework.

Application Class	Description	Compared Against
Document	In this class the candidate does not have access to external information such as schema. The encoded instances that it produces are still self-contained, but may perform various document-analysis operations such as frequency-based occurrence compression.	Gzipped XML
Schema	In this class the candidate does not perform any manner of document analysis but may rely on externally provided information, typically a schema, and the resulting bit stream may not be self-contained.	Plain XML
Both	In this class the candidate uses methods available in both of the Document and the Schema cases.	Gzipped XML
Neither	In this class the candidate has no access to external information such as a schema, but the encoded instance that it produces are still self-contained, and does not perform any compression bases on analyzing the document. Typically, simple tokenization of the individual XML document is performed.	Plain XML

Table 14. W3C Binary XML Framework Test for Application Classes for XML Structure (From W3C, 2007)

3. Testing Framework - Drivers

The execution of the test-corpus by each candidate was conducted using Japex to efficiently enable the testing of the large set of documents consistently by each candidate technique. Note that Japex is a simple open source Java-based tool used to write micro-benchmarks that enable consistent measurements across test cases (Japex, n.d.). Japex can be freely downloadable from <https://japex.dev.java.net/>. The entire test-corpus of XML documents and the Japex drivers for each candidate technique can be freely downloaded from <http://www.w3.org/XML/EXI/#TestingFramework>. However, not all of the technique engines are not included in this download due to proprietary code and or

licensing restrictions. Additional guidance on the framework set up and how to acquire the candidate technique engines can be obtained from the framework release notes at site http://www.w3.org/XML/EXI/framework/RELEASE_NOTES.txt.

Two common variance-inducing conditions within the testing environment were considered in the experiment and actions taken to mitigate their effects to achieve fair and standardized results:

- Implementation code language – C++ and Java are the dominant candidate implementations languages. C++ and Java are somewhat similar procedural languages, but have differences in performance characteristics; C++ compiles directly to the architecture hardware and can produce faster execution times than a Java implementation given Java is a Just-In-Time (JIT) Virtual Machine (VM) language.

Notionally, once a VM, such as the Java VM (JVM) is warmed up, in active memory and the Just-in-Time compiler has interpreted and compiled the code, the VM's performance mirrors that of C++.

Because the first run of an application generally does not reflect true application run-time performance due to application warming up requirements, many runs were conducted until measurements were stable.

- Networking and system start up – In order to discount the effects of external networking, cases were ran local (loopback) and across network connections. Of course the test-corpus cannot run across every possible network connection configuration so absolute results for a particular network configuration remains implementation dependent.

As of this thesis, the Japex drivers are being revised for the next phase of the standardization process: the interoperability testing of multiple code implementations of the same technique, EXI. The activity of the revisions of the drivers and interoperability testing can be obtained from <http://www.w3.org/XML/Group/EXI/TTFMS/>; however, this requires W3C password-protected access. Membership is relatively easy to obtain,

simply submit an e-mail request to the EXI mailing list asking to participate. It is best to have an affiliation with an existing W3C member or else meet the W3C requirements to become an Invited Expert.

4. Testing Framework - Candidates

The *Efficient XML Interchange (EXI) Measurements Note* tested nine different candidates for consideration as the W3C recommendation for continued development into standardization (W3C, 2007). This document lists references and links to the candidate techniques as necessary and can also be found from the XBC workshop discussions (W3C, 2005). The following sections summarize the candidate techniques (W3C, 2007).

a. X.694 ASN.1 with BER

The Abstract Syntax Notation 1 (ASN.1), from the International Telecommunications Union - Telecommunications Group (ITU-T), and the International Standards Organization (ISO), is a schema much like a XML Schema for describing abstract message types (W3C, 2007).

The ITU-T/ISO Basic Encoding Rules (BER) is a set of encoding rules used with ASN.1 that produce a binary representation of a set of values described by the ASN.1 schema.

The ITU-T/ISO X.694 standard provides a mapping of XML Schema (XSD) to ASN.1, and allows the use of ASN.1 and its associated encodings in XML applications.

For example, <tag>textual content</tag> is replaced by a similar binary encoding consisting of a tag-length-value descriptor. Efficiencies are gained from two properties of this format:

- 1) The tags and lengths are binary tokens that are in general much shorter than XML textual start and end tags.
- 2) The content is in binary instead of textual form.

Advantages of X.694ASN.1 with BER are its similarities to the XML tagging format and this encoding can be five to ten times more compact than native XML. The process is efficient requiring no special compression or other CPU intensive algorithms. Additionally, this method is mature, stable, and well studied.

Disadvantages are that it requires an XML schema to work, and there are some functions of the XML Information Set cannot be represented by this technique, for example, comments and XML Processing Instructions.

b. X.694 ASN.1 with PER

This encoding is similar to X.694 ASN.1 with BER except it allows direct output in textual XML, and has a slightly more compact binary encoding. Packet Encoding Rules (PER) is designed to minimize the size of messages between machines in limited-bandwidth environments. Its origin of effort was efficient air-to-ground communications for commercial aviation, and has seen implementation in cell phones, internet routers, satellite communications, internet audio/video, and many other low-bandwidth areas (W3C, 2007).

c. Xebu

The Xebu format is designed for XML on small mobile devices with the following design goals: stream-able, small footprint and low implementation cost. Xebu uses events similar to StAX and SAX, and maps those strings to small binary tokens with the mappings being discovered as the document is processed (W3C, 2007). If a schema is available, three options are available:

1. Pretokenization—This populates the token mappings beforehand based on the strings appearing in the schema.
2. Typed-content encoding—This resulted in a more efficient binary form for certain datatypes than can be achieved without a schema.
3. Event omission—This leaves out events from the sequence if their appearance and placement can be deduced from the schema.

Key desirable Xebu features are its ability to work with or without a schema, direct XML mapping, and relatively straightforward implementation.

The Xebu implementation used during testing was optimized for mobile-phone applications, and did not perform as well as an implementation written for desktop machines or servers. Additionally, schema cases were run only with pretokenization enabled. Typed-content encoding was not enabled. The event omission option was not able to handle all schemas, so only a subset of the test document schemas produced results.

d. Extensible Schema-based Binary Compression (XSBC)

Extensible Schema-based Binary Compression (XSBC) encodes a XML document based on its schema into a binary format that is more compact and faster to parse than textual XML. Based on the schema, lookup tables are created with n-bit indexes to entries. Once the tables are created, the XML document is translated from XML to XSBC (W3C, 2007 & Norbraten, 2004).

Advantages of XSBC's are its simplicity and nearly obvious first step to XML compression. The main disadvantage is it requires a schema. However, the XSBC team noted that it is possible to develop a schemaless XSBC, by limiting all datatypes to string.

e. Fujitsu XML Data Interchange Format (FXDI)

The Fujitsu XML Data Interchange (FXDI) format design goals are document compactness with fast decoder and encoder programs, which run with a small footprint without involving much complexity (W3C, 2007).

FXDI is based on the W3C XML Schema Post Schema Validation Infoset (PSVI) using the Fujitsu Schema Compiler to compile W3C XML Schema into a "schema corpus." A schema corpus contains all the information expressed in the source XML Schema document plus certain computed information such as state transition tables.

FXDI works well with conventional XML document redundancy-based compression such as GZip, which facilitates use cases that need additional compression and that can afford to spend the additional CPU cycles.

*f. **Fast Infoset (FI)***

Fast Infoset (FI) is an open, standards-based binary format based on the XML Information Set ITU-T Rec. X.891 | ISO/IEC 24824-1. Fast Infoset documents are generally smaller in size with faster parsing and serialization than equivalent XML documents. FI is designed to optimize compression, serialization and parsing, while retaining the properties of self-description and simplicity. The use of tables and indexing is the primary mechanism for FI compression (W3C, 2007).

Within FI, it is possible to selectively apply redundancy-based compression or optimized encodings to certain fragments. Using this capability, as well as other advanced features, it is possible to tune the “sweet spot” of compression emphasis for a particular application domain. Uniqueness of the Fast Infoset approach include:

1. The Schema and application classes are not schema optimized.
2. If namespace prefixes do not need to be retained, additional reductions in compactness and processing efficiency are possible.
3. Allows for restricted alphabets to achieve a more-optimized binary format.

*g. **Efficient XML Interchange (EXI)***

Efficient XML is described in detail in Chapter VIII.

h. X.694 ASN.1 with PER + Fast Infoset

This approach uses both X.694 with PER as previously described, as well as Fast Infoset (FI) when applicable. Where there is an XML Schema, X.694 is used to map the schema to ASN.1 (W3C, 2007). If there is no schema or if the XML document deviates from the schema, the entire XML document is serialized using Fast Infoset (FI).

i. Efficiency Structured XML (esXML)

Efficiency Structured XML (esXML) is a flexible, compact, and efficient compression process that uses a Hybrid Byte-Aligned Format (HBAF), which is a simple, low-level mechanism in the memory access layer to pack bit-aligned data within byte-sized data (W3C, 2007). Encoding of datatypes is flexible, supporting text-based values, as well as schema-informed binary encoding of defined datatypes. The unique feature of esXML is its ability to utilize many different external forms to define the data, not restricted to a traditional XML schema though the use of an XML Meta Structure (XMS), where an XMS is a document template that can be referred to later within a document.

The motivation for esXML is to support the need for pointers, deltas, and random access. It accomplishes these tasks by the addition of a storage layer called Elastic Memory. This memory directly supports efficient processing of pointers, low-level deltas (i.e., byte/bit oriented ranges), and random modification (inserts, deletes, replacement) without parsing and serialization.

The evaluated version of esXML did not implement schema-informed encoding, dropped tests for some documents, and produced debugging information, which drastically affected processing efficiency. During testing it operated in byte-oriented mode and not the more compact bit-oriented compact form.

5. Testing Framework - Results for Compactness

After full execution on every candidate with every test-corpus document, the most compact candidates were, in order, EXI, FXDI and FI:

- Efficient XML Interchange (EXI) was the best performer for nearly all tests.
- FXDI is the next best candidate, but only when it leveraged a schema.
- Fast Infoset (FI) was third, but did not come close to the performance of EXI or FXDI.

For the Military, Scientific, and Storage use groups the availability of a type-aware schema is essential for achieve truly substantial gains in compactness. Interestingly, the Military was the only use group that had schemas for a majority of the test documents.

Cases without a schema tended to approximate the effect of GZip. Best schemaless results came from the Document and Storage use groups, with Finance and Scientific also getting significantly better than GZip performance.

The necessity for a schema become more apparent when the XML documents contain low amounts of redundant data, which is exacerbated when there is also little structure in the XML document. It was noted that without a schema, a highly structured document can achieve fair results, but only slightly better than GZip.

The overall conclusion is that Efficient XML Interchange (EXI) consistently delivered better results than the other candidates for both schema-informed and schemaless document types. Table 15 lists the comparison of results for FXDI and EXI, the top two techniques in terms of their improvements over the baseline comparison tests, where the higher the percentage the better.

	Neither	Document	Schema	Both
Efficient XML	70%	10%	80%	10%
FXDI	60%	0%	70%	0%

Table 15. W3C Binary XML Framework Test Results Summary of Percentage of Improvement for Compactness Over Baseline (From W3C, 2007)

6. Testing Framework - Results for Processing Efficiency

Processing efficiency results have the largest variance of all the tests performed. This is as expected given performance efficiently is highly dependent on implementation approach and usage of code optimization techniques, or lack thereof. However, two candidates stood-out without a decisive winner: FXDI and EXI. Table 16 lists the comparison of results for FXDI and EXI in terms of their improvements over their baseline comparison tests; the higher the percentage the better.

The application use groups Finance, Military, and Storage achieved the best performance efficiency results, and each was better than GZip. Schema usage did not increase performance, counter to initial expectations, though schema-informed is essential for optimal compression.

	Neither	Document	Schema	Both
Efficient XML	40% encoding 180% decoding	-10% encoding 70% decoding	Slightly less	No difference noted
FXDI	50% encoding 160% decoding	20% encoding 110% decoding	Slightly less	No difference noted

Table 16. W3C Binary XML Framework Test Results Summary of Percentage of Improvement for Processing Efficiency Over Baseline (From W3C, 2007)

7. Testing Framework - Results for Round-Trip Conversions

Round-trip tests were conducted on every test-corpus case by each candidate technique. The round-trip decoded XML document is compared to the original input XML document with a differencing process that supported all the fidelity options and

utilized PSVI criteria to be XML structure aware, e.g., aware of XML attribute and element order variances that are allowed. The results of this test are a simple pass or fail; the round-tripped document either matched the original or did not.

The conclusion of round-tip testing was that the candidates Xebu, FXDI, Fast Infoset, and EXI passed all tests with the other candidates receiving at least one failure.

C. EXI SELECTION AND BASELINE (GZIP) TESTING

Given the results from the W3C framework testing, EXI was the only technique that stood out in all three measurements, EXI was therefore selected as the candidate technique for further study and possible standardization into the XML stack.

Since GZip is the traditional method to compress XML, as well as other text-based files, a full EXI-to-GZip benchmark test was then conducted evaluating both in terms of size, efficiency, and W3C use case property demands compliance. Similar to the candidate selection testing, the baseline testing is also defined within the W3C *Efficient XML Interchange Measurements Note* (W3C, 2007) and evaluation results from *Efficient XML Interchange Evaluation* (W3C, 2008).

1. Compactness Comparison

One of the W3C's property goals from the XBC effort was to produce a general non-domain-specific algorithm that delivers a compact, less-than-original XML document, binary XML output. In other words, the algorithm must always, and for all cases, deliver a compressed file size that is less than the original file. If any technique ever delivers results over 100% of the original raw XML file, then that technique should be discarded, even if such occurrences are rare. This does not mean the selected candidate had to always outperform the other techniques just that it has to always deliver a result file less than the original input XML document. Of course, the best candidate should more often than not deliver a result file superior to all other candidates in addition to always being less than the original input XML document. Figure 23, sorted in

increasing order of EXI percentage of original document $\left(\frac{EXIsize}{XMLsize} \right)$, depicts the results

of the GZip and EXI comparison test results for the test-corpus of documents; EXI is the blue line, GZip is the pink line, and the raw text XML document is the red line.

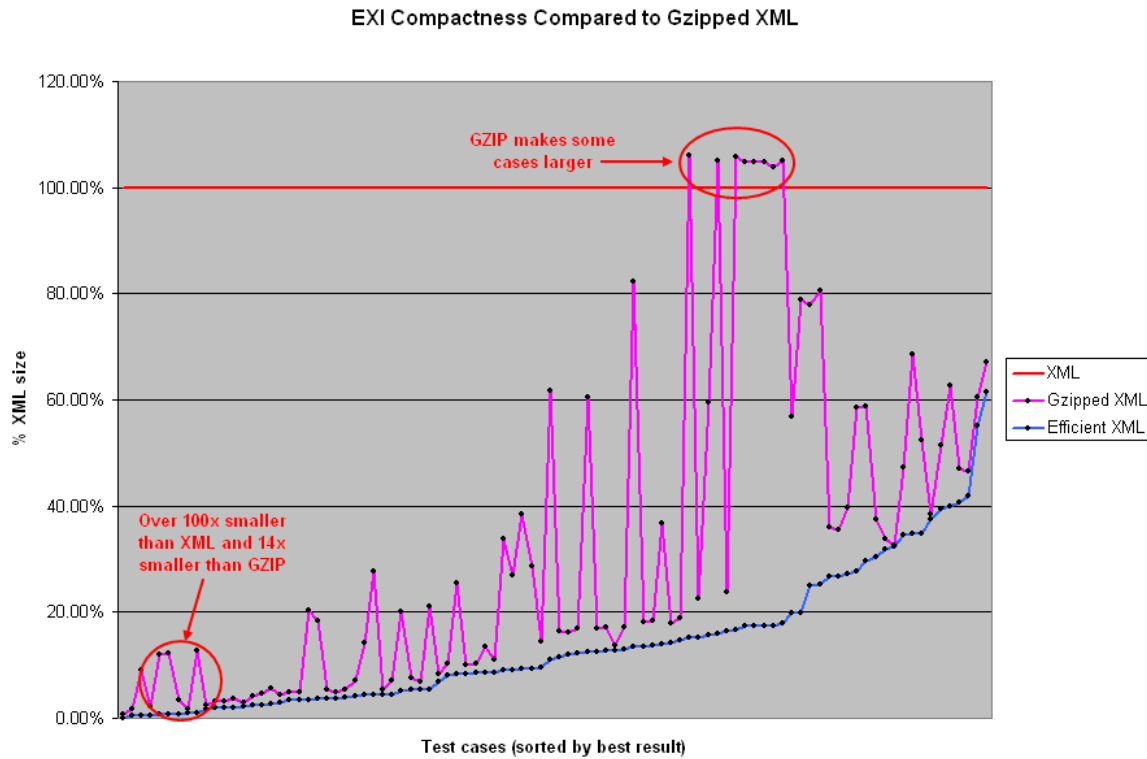


Figure 23. EXI Compactness Comparison to Traditional GZip (From W3C, 2008)

A summary of the results of the baseline compactness comparison:

- EXI at worst is equal to GZIP, though in general is a more compact representation of XML.
- EXI scales better than GZIP as documents grow in size.
- EXI was smaller than the original XML document size for every case, while GZip had a number of cases that exceeded the original document file size.

2. Property Comparison

A comparison of EXI and GZip in terms of the W3C XBC working group required properties are summarized in both Table 17 and Table 18 (W3C, 2008).

XBC Property	GZip		EXI	
Directly Readable and Writable	No	Requires the creation of an intermediate file	Yes	EXI is a dynamic event drive technique with support API (SAX, and DOM)
Transport Independence	Yes		Yes	
Compactness	No	Cannot take advantage of Schema and does not deliver comparable EXI results	Yes	
Human-language Neutral	Yes		Yes	Is XML-based
Platform Neutral	Yes		Yes	
Integratable into the XML stack	Yes		Yes	A development requirement
Royalty Free	Yes		Yes	Is a W3C open standard
Fragmentable	Yes		Yes	Can represent any fragment
Streamable	Yes		Yes	
Round Trip support	Yes		Yes	Supports lossless and lossy
Generality	No	8/20 (Table 19)	Yes	19/20 (Table 19)
Schema Extensions and Deviations	No		Yes	Can optimize using the schema
Format version identifier	Yes	XML file and ZIP header	Yes	EXI Header
Content type management	Yes		Yes	
Self-contained	Yes		Yes	With or without schema capable

Table 17. Comparison of W3C Binary XML Property Requirements Between GZip and EXI (From W3C, 2008)

Property	GZip		EXI	
Processing Efficiency	Prevents		Does Not Prevent	Both memory footprint and speed are better
Small Footprint	Does Not Prevent		Does Not Prevent	
Widespread Adoption	Does Not Prevent	Is widely used	Does Not Prevent	Is an open standard tailored for XML
Space Efficiency	Prevents		Does Not Prevent	
Implementation Cost	Does Not Prevent		Does Not Prevent	
Forward Compatibility	Does Not Prevent		Does Not Prevent	

Table 18. Comparison of W3C Binary XML Property Demands Between GZip and EXI (Must Not Prohibit) (After W3C, 2008)

EXI is able to meet nearly all of the W3C properly demands for a compact binary XML format. GZip in a line-by-line comparison does not indicate it is a terrible consideration, but in terms of compactness, EXI excels.

GZip's success is from its wide acceptance in the IT world as the standard compression technique, much like the way XML achieved its success as the standard data exchange format. However, unlike XML's success story, for GZip, there is a better compression algorithm for the XML family of technologies, and that is EXI.

3. Generality Comparison

Similar to the property comparison, the overall generality results of EXI-to-GZip-to-XML itself is listed in Table 19, where an ☒ indicates compliance and an empty cell is not compliant. Out of the 20 general comparisons, EXI was able to meet 95% of the criteria (19/20); namely all criteria except for exact preservation of whitespace formatting, which by definition of PSVI rules is allowed to convey significant information.

Criteria	XML	GZip	EXI
Can represent documents without a schema	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Can represent documents that include elements and attributes not defined in the associated schema (i.e., open content)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Can represent any schema-invalid document	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Can leverage available schema information to improve compactness, processing speed, and resource utilization			<input checked="" type="checkbox"/>
Can leverage available schema information to improve compactness, processing speed, and resource utilization even when documents contain elements and attributes not defined in the schema			<input checked="" type="checkbox"/>
Can leverage available schema information to improve compactness, processing speed, and resource utilization for any schema-invalid document			<input checked="" type="checkbox"/>
Can leverage document analysis to improve compactness		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Can suppress document analysis to increase speed and reduce Resource utilization	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>
[optional] Can adjust document analysis to meet application performance and resource utilization criteria		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Can structure the binary XML stream to increase net compactness when off-the-shelf compression software is built in to the communications infrastructure			<input checked="" type="checkbox"/>
[optional] Supports high fidelity XML representations that preserve an exact copy of the original XML document, including all whitespace and formatting	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Supports reduced fidelity XML representations that preserve all data model items, but discard whitespace and formatting to improve compactness	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Supports reduced fidelity XML representations that preserve all information needed by a particular application, but discard specified information items that are not needed (e.g., comments and processing instructions) to improve compactness	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Supports reduced fidelity XML representations that preserve the logical structures and values of an XML document, but discard lexical and syntactic constructs to improve compactness	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Can consistently produce XML representations that are close to the same size or smaller than XML documents compressed using GZip		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Can consistently produce more compact XML representations than XML documents compressed using GZip			<input checked="" type="checkbox"/>
Can consistently produce more compact XML representations than binary XML documents created with document analysis suppressed, then compressed using GZip			<input checked="" type="checkbox"/>
Can consistently produce XML representations that are close to the same size or smaller than the equivalent ASN.1 PER encoding plus 20%			<input checked="" type="checkbox"/>
Can consistently produce XML representations that are more compact than the equivalent ASN.1 PER encoding plus 20%			<input checked="" type="checkbox"/>
[optional] Can consistently produce XML representations that are more compact than the equivalent ASN.1 PER encoding plus 20% compressed using GZip			<input checked="" type="checkbox"/>
Totals	8	10	19

Table 19. Binary XML Generalized Comparison of EXI and Gzip
(From W3C, 2008)

D. EXI USAGE RECOMMENDATIONS AND LIKELY IMPACT

Often the question with new technologies is how should the technology be used, when should it be used, and what will be the expected impacts of the new technology. The W3C working group developed two documents to assist decision makers in assessing such concerns titled *Efficient XML Interchange (EXI) Best Practices* (W3C, 2007) and *Efficient XML Interchange (EXI) Impacts* (W3C, 2008).

1. Domain Applicability

Because EXI is centered on the traditional XML Infoset, it is applicable to any XML domain, but can deliver varying degrees of efficiency and compactness based on each domain's specific design characteristics (W3C, 2008). EXI's is applicable to any domain whenever native XML cannot support the domain's efficiency and compactness requirements. However, EXI does not need to be implemented within domains that do not benefit from its strengths.

Domain-cases that are likely to see the least improvement from EXI are those that use unique characters or custom octets such as base64 characters.

2. Human Readable

To address the demand to retain the human-readable format property of XML, the recommendation is to let a tool do the translation transparently to the user. Few people truly edit XML documents outside of an XML-specific editing tool environment. Therefore, let the editor read and write EXI, but present traditional XML text to the user of the editor tool. Retain the human readability of the underlying XML file by enabling the tool to switch between XML and EXI. While the user loses the ability to use a plain-text-editor, they can still use an XML-specific editor. This is not a handicap if the compressed EXI document remains ungarbled and correctly formatted.

3. Domain Optimization

Depending on the implementing domain, pruning options can be used to eliminate excessive and unnecessary information from an XML document. This approach increases efficiency and delivers a more compact file. Often a domain is only concerned with specific portions of a document, and all the metadata and header information is syntactically irrelevant, though required for the XML structure or the human-readability. EXI content optimization through fidelity options enables higher efficiency and compactness; however, this comes at the cost of lossy compression.

4. Security and Signature

The method to secure information across the Web is digital signature and encryption. EXI is capable of supporting both without modification, but with a few EXI-specific warnings depending where within the signature-encryption chain EXI is applied.

a. Output Alignment

If EXI is implemented before either the signature or encryption of the XML document, and assuming the EXI results are placed within another XML document, the EXI encoding options for byte-alignment must be set because both signature and encryption operate on octets (bytes). As shown in Figure 24, if EXI is byte-aligned, the EXI compression technique can be applied on fragments of XML in-between signature and encryption (Williams, 2009).

However, if the XML document is compressed with EXI after signatures and encryption, then the alignment of EXI output is not of concern. Both XML-signature and XML-Encryption output valid XML, although with blocks of base64 data. All EXI requires for operation is a valid XML input document, which both signature and encryption produce. However, signed and encrypted XML documents will not compress well with EXI due to the high volume of pseudorandom high-entropy base64 data.

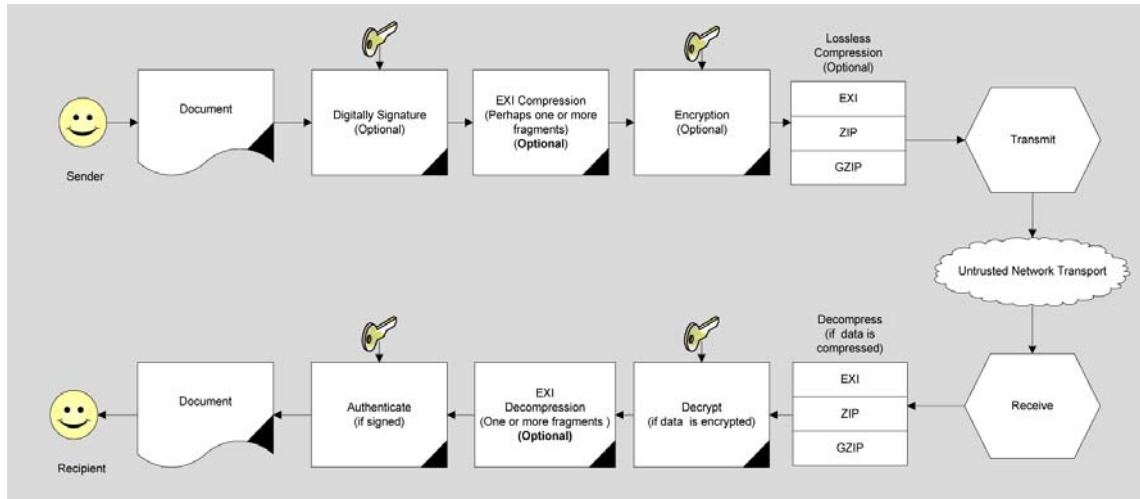


Figure 24. EXI Integration with XML Encryption and Signature (From Williams, 2009)

b. XML Signature

Digital signature operations require a predefined (sender and receiver agreed) canonicalization of the XML document. The XML specification for digital signature requires the inclusion of a canonicalization method element as part of the metadata describing the signature, and is used to verify which XML Infoset items it can ignore and how it should handle namespaces. Since the XML canonicalization methods are identified by URI (namespace), the EXI specific options of Preserve.pis, Preserve.prefixes, and Preserve.lexicalValues, and optionally Preserve.comments are required to be set to “True” before encoding XML to EXI for follow on digital signature.

c. XML Encryption

Encryption requires the data to be removed from the Infoset prior to encryption. This is because the first occurrence of each string under EXI is output to the EXI file in raw ASCII format. XML defines procedures to replace elements and their content with a new element containing a cipher and metadata describing how the cipher was generated. This replacement process transforms the XML into octets, in base64. Often, though not required, a XML canonicalization method is used to produce these octets to reduce the amount of contextual information being encrypted.

5. Domain Integration

a. Web and HTTP Servers

EXI is optimally suited for Web-content and Web services within low-bandwidth environments to enable XML to domains previously unable to support native XML. Ed Day (2007) of Objective Systems Inc, a developer of open standards that promote the interoperability of systems was quoted, “We are trying to expand the Web by getting XML into places where it could not be used before.” This goal is most widely accomplished by deploying implementations of EXI into HTTP server environments.

b. XML Modifications Considerations – Schema Mandate

EXI is designed operated on the XML families of technologies without modification or disruption to the XML already in existence. However, with the future in mind, and optimization taken into consideration, certain XML infoset modifications might enhance the impact of EXI, such as a descriptive schema requirement for all XML documents.

c. Initial EXI Distribution

Because the adoption of EXI will initially be sparse, care must be taken to ensure data exchanges in the EXI format are between EXI enabled systems. To ensure interoperability during initial EXI adoption, it may be prudent to exchange both native XML formatted documents as well as EXI versions whenever EXI compliance between systems is uncertain.

E. CHAPTER CONCLUSION

EXI was not arbitrarily recommended by the W3C for standardization; it was thoroughly tested against numerous criteria and techniques. Each time EXI proved to be the most optimal choice. EXI has proven to be a valid and preliminarily verified solution that can meet the stringent requirements of the W3C and the XML Binary

Characterization (XBC) working group. Compared to all other candidates, EXI proved to have the best compactness, efficiency, and accuracy for a standardized binary XML format.

F. CHAPTER SUMMARY

This chapter discusses the testing framework used by the W3C to evaluate candidate compact binary XML formats. The nine candidate techniques are discussed along with their summary of results, with the Efficient XML Interchange (EXI) technique being the superior and chosen method for continued standardization development. EXI is further analyzed by comparison with the industry standard GZip technique. The chapter concluded with the W3C's preliminary recommendations for EXI implementation and usage.

VII. OPENER-EXI IMPLEMENTATION RATIONALE

A. INTRODUCTION

This chapter discusses the administrative constraints and challenges an EXI implementation will face in terms of licensing, deployment, and DoD specific integration constraints. Specific examples throughout this chapter are made in regards to OPENER-EXI, the NPS open source partial implementation of the EXI specification. The goal of this chapter is a set of recommendations for EXI development and deployment that will deliver the deepest and fastest adoption of OPENER-EXI or other EXI solution into existing network architectures.

B. LICENSING

According to Webster's Online Dictionary, licensing is "a permission granted by competent authority to engage in a business or occupation or in an activity otherwise unlawful" (License, 2009). Licensing in software specific terms is an agreement between a software creator and user of the software defining what the user can and cannot do with the software source-code, and potentially, how the software may be used (Kayne, 2010). This is commonly called the End User License Agreement (EULA), that check box that appears when installing software that says "click here if you agree," which is more often than not simply overlooked.

Software licensing comes in many flavors depending upon the software creator's desires, from the loose to the restrictive, but all licenses are in place to specify the copyright of the software in terms of duplication, modification, and distribution. There are dozens of formal licensing agreements and countless custom versions of each, making software licensing a complex and challenging aspect of software development and deployment.

In a high-level overview of software licenses there are arguably three general types of licenses, each with their pros and cons for the user and developer: Open, Free, and Proprietary.

1. Open Source

Open source software (OSS) is the most liberal of the licensing formats as it allows users full access to the source-code (Beard, 2007). Often this type of software is developed collaboratively among many contributors that share a common interest in the software. Individuals that have a vested interest in the development of the software work together on a software project free of charge. The intent of such efforts is to make the software adaptive to user specific desires, which makes the software better than market code because it meets all the needs of user in both executions an interaction (Nordquist, Petersen & Todorova,2003).

a. OSS Origins

Richard Stallman and the Gnu's Not UNIX (GNU) project first presented the OSS concept, focused on building a Unix-like operating system free of charge. This OSS initiative started by GNU changed the rights of users as they pertain to the use of the software source-code (Beard, 2007). From the GNU (2009) website they coined the phrase "...you should think of 'free' as in 'free speech' not as in 'free beer,'" making the following declarations:

0. The freedom to use programs for any purpose.
1. The freedom to study how programs work, and adapt them to your needs.
2. The freedom to redistribute copies so you can help your neighbor.
3. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits.

b. OSS Repackaging

Part of the OSS methodology is OSS is free to use as seen fit, which includes the repackaging and selling of the software. This implies someone could download an OSS package, and then turn right around and sell that same software for profit. This is not contrary to the OSS philosophy, but may seem unnecessary or rude.

Repackaging of OSS software for commercial profit is common. The logical question is why would someone pay for what they could get for free? The answer, people are willing to pay in some cases in order to get technical support for the software, where the OSS “free” software comes with no warranties or technical support. These sold packages, OSS or not, are typically called Original Equipment Manufacturer (OEM), software distributed under a company name and license.

It is typical to think of OSS as free, though most OSS licenses provide developers the right to charge a small nominal fee for repackaging and redistribution. However, this fee is trivial compared to the typical commercial cost of producing software.

c. OSS Copy-Left

To prevent the software from being closed, many OSS package contain a “copy-left,” which forces anyone who redistributes OSS packages to do so on the condition the redistributed package continues to remain free and open under the same OSS license (Laurent, 2004).

d. OSS Viral

“Viral” is another common term within OSS licensing with GNU Public License (GPL) being a common viral licenses. Viral is often a label of criticisms of OSS licenses, such as in GPL, because it prevents “viral” licensed programs from linking to other libraries or programs that are not of an equal license (Laurent, 2004). This can be

seen as preventing the OSS vision by preventing uninhibited free growth of software because it denies arbitrary software from combining into a larger whole depending on licensing agreements.

Richard Stallman, the GPL founder, considers the “viral” label as both offensive and wrong (Williams, 2002):

Software under the GPL never "attacks" and "infects" other software. Rather, software under the GPL is like a spider plant: If one takes a piece of it and puts it somewhere else, it grows there, too.

e. OSS Examples

There are many examples of OSS licenses that have fewer commercial restrictions compared to GPL: Apache, Berkeley Software Distribution (BSD), Lesser General Public License (LGPL), Massachusetts Institute of Technology (MIT) and Mozilla Public License (MPL). Each of these licensing methodologies allows for some forms of commercialization of the OSS code after being modified. A quick summary comparison of some common OSS licenses, in regards to the limits and freedoms they impose on the users and developers of OSS packages is contained in Table 20.

License Type	Notice Of Modifications	Redistribution Rights	Must Retain Original Code	Link To Original Code	Library Notice
Apache	Yes	Yes	No	No	No
BSD	Yes	Yes	No	Yes	Yes
GPL	Yes	Only under GPL or LGPL	Yes	No	Yes
LGPL	Yes	Only under GPL or LGPL	Yes	Yes	Yes
MIT	Yes	Yes	Yes	Yes	Yes
MLP	Yes	Only under MLP	Yes	Yes	Yes

Table 20. Comparison of End-User Rights for Common OSS Licenses (After Laurent, 2004 & Beard, 2007)

f. OSS Conclusion

OSS has enabled the creation of a mass of software that is free of charge for users that is often superior to commercially available products because the code is built and maintained by the users, for the users. The end-state of OSS is that users can do what they want with the code as they see fit, with the belief that the result can become the best software possible; those who need and use it, build it.

2. Free Source, Share Source, Shareware

Free software is often misunderstood. The software is free for use, or available for a nominal contribution fee, but unlike OSS licenses, it might not permit users to view, modify or sometimes even to redistribute the code. Generally, this implies a single-user license with limited to no technical support. Common well-known examples of this type of software are the Sun Industry Standard source License and the Academic Free License. It is common to hear the term shareware when discussing free source. The code or complete application package is available for initial use without cost, but contributions are expected for commercial or extended use.

These types of licenses enable the original developers to retain control over the code, and the direction of the implementation base. Reasons for such licenses as opposed to the OSS models can include code complexity and the desire to ensure a certain level of quality and direction in their named package. A well-known application example would be Adobe Reader, which is freely distributed but without source-code. Because it is a good product, Adobe Reader's licensing plan has made the .pdf format evolve into a defacto industry-recognized format for document distribution. Other examples of free software are personal-use utilities that have single-focus missions such as disk management tools, and numerous multimedia players such as iTunes from Apple. These applications are free to use, but the source is restricted by software licenses, trade-secrets, patents, or copyright.

Free Source may thus sometimes be free for use, but not for modification. Such free distribution enables a wider distribution base while keeping secure the proprietary source-code (Michaelson, 2004).

3. Proprietary and Commercial

Proprietary or Commercial software is anything that has to be purchased beyond a nominal fee for use. This type of software does not permit users to view, modify or redistribute the code. Well-known examples of this are Microsoft's Office suite and their operating systems. They are only legally available through purchase, and have limitations on their use, such as single installations, or single license per computer.

The benefit of such software is that it often comes with a warranty and support, but does not allow users flexibility in application development for specific needs outside of the licensed stipulations. Microsoft products do come packaged with Visual Basic, a programming interface that allows for some customization of the suite, but the core competencies of the suite are not directly accessible.

4. General Licensing Considerations

The choice as to which type of license to put on a software solution depends on a number of factors:

- Is the code developed around proprietary secrets
- Is it an essential part of the business model
- Is the intent to become an industry standard
- Is the focus to make the solution widely adopted

Microsoft, for example, has built their business model around their software solutions. If Microsoft had made their code freely available, that would have eliminated their source of revenue. On the other hand, the Adobe Reader example had the intent to make a common format to standardize the way documents are shared, and to create some consistency between documents and printers. Making the Adobe Reader solution free

thrust Adobe Reader into the marketplace, and in turn, created increased demand for their other proprietary applications. Another case example is BSD Unix and their efforts to create a solid operating system that is flexible to adapt to anyone's needs. Making the BSD Unix source-code free and open enables the operating system to achieve widespread use with constantly evolving improvements, which arguably has delivered the most stable general operating system available in the world (Laurent, 2004).

The end choice of license depends on intent of the solution as well as the business model under which the source-code is developed. Table 21 summarizes the 6 general license types in use in terms of software development considerations.

License	A.K.A.	Permissions Granted To Users
GPL	Viral	The code is free, but must remain free
LGPL	Free and free	Code is free and portions are convertible to another license type
BSD/MIT/Apache	Free with credit	The code is free to use as seen fit such as convert to another license in full as long as the original code is given credit
Share Source	Use not see	Can use the code only, but its free or nominal fee
Proprietary License	Pay and See	Pay for the use and view of the source
Proprietary Closed Source	Pay and not see	Only have access to the application binaries

Table 21. Developer Perspective Licensing Considerations (After Laurent, 2004 & Michaelson, 2004)

5. OPENER-EXI Licensing Considerations

The primary motivational goal for EXI is to bring to the WWW and other networks a better XML interchange to further XML Web penetration, specifically to small mobile and handheld devices. Making an EXI solution non-viral OSS encourages adoption by both individual developers and well-established corporations due to the reduced development overhead, "take this working source-code and use it as you see fit,"

without having to invest extensive time and effort into development. Using a license that encourages broad use can help meet the goals of EXI to make it an adopted and utilized industry binary XML standard.

The OPENER-EXI solution is best fitted with an open and free license (such as Apache or LGPL) to increase the expected likelihood of widespread adoption. At the same time this grants corporations the right to customize the OPENER-EXI solution and package it into their existing products, as they see fit, for profit. Placing a non-viral free license on the OPENER-EXI code allows it to be used without restrictions with proprietary source, which should encourage the corporations to adopt the solution into their codebase. This in turn helps to deliver a wider dissemination of EXI solutions.

As of this thesis, the OPENER-EXI project is licensed under the Apache version 2.0 license, which is open, but does require the redistribution of OPENER-EXI to be cited with credit (Apache, 2004).

C. OPENER-EXI IMPLEMENTATION CONSIDERATIONS

EXI's focus is XML interchange, which is most common within the WWW environment. Often the only method to exchange data unilaterally is via port 80 because port 80 is one of the few always open ports. Most other ports are turned off to prevent potential security leaks, but port 80 is almost always open to allow for Webpage serving. Port 80 data transfers are the only assured medium that allows for the widest uninhibited file exchange regardless of sender and receiver across the WWW. For example, go to any Web page and you can download content from that page with a right-click of the mouse and select "Save Target As..." from the pop-up menu. Such action is conducting a data transfer via port 80.

The alternative is to use a custom port for a standalone application. An example of this is iTunes, which uses a port other than 80 to download music and other content. Unless the other port used by iTunes is open along the entire path from sender, Apple, and receiver, your desktop, the transfer cannot be completed. While in most cases the iTunes port is open, but for a network port in general this assumption of an unobstructed complete path cannot be made.

In the business world or DoD settings, security reasons prevent most ports other than 80 from being open, leaving port 80 as the only assured open path to conduct file transfers. Port 80 has been used with HTTP tunneling to exchange data of all sorts, to include chat and other high volume communications, and is expected to remain a persistent avenue for WWW communications in the future.

1. Web Integration Deployment Focused

In view of the port 80 dilemma, the initial OPENER-EXI implementation push is to embed it within Web servers and browsers. This enables seamless and rapid utilization of EXI below the application layer of any network. Enough network traffic, e-mail being a likely number one candidate, is XML-based, which justifies the incorporation of EXI into the Web server environment. Building a negotiable EXI compression technique for data transfer reduces both bandwidth and processor utilization.

2. HTTP Negotiated File Format Transfers

Web servers exchange data files using a number of compression techniques that are negotiated between a client, Web browser, and servers before data transfer execution: GZip, raw format, and eventually EXI. Figure 25 depicts a notional HTTP transfer with compression negotiation based on the governing process of RFC 2616.

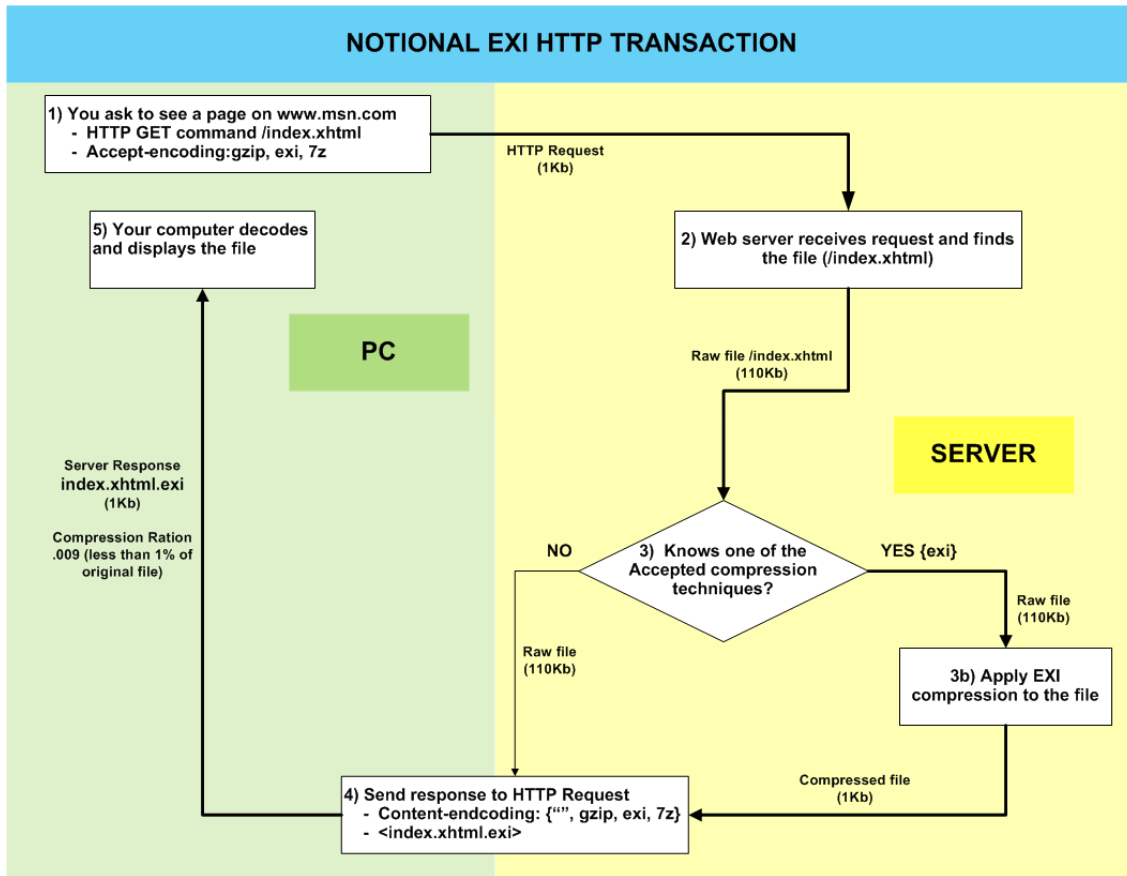


Figure 25. EXI as a Negotiable Compression Technique at the Web Server

Adding EXI as an option for http content negotiation by Web servers and client will be an important new capability to achieve. An additional caveat to embedding into Web servers is that as EXI's popularity grows, the demand for EXI-capable Web browsers will grow, which will encourage even broader EXI deployment.

Ultimately, EXI integration in servers and browsers enables a broader network deployment, enabling devices incapable of handling large XML files to efficiently process and transfer XML. Low-bandwidth high-volume handheld and mobile devices can then receive a wider variety of network content to include detailed graphics, M&S files, and nearly any other application format in use given the likelihood those files are XML based.

3. Apache Server Considerations

Apache, a free Web server application, is, and has been the dominant market shareholder of Web servers in the world (Netcraft, 2009). As of April 2009, Figure 26 depicts the historical overview of worldwide Web server market shares for a number of Web server distributors. From this figure it can be seen that Apache currently holds nearly 50% of the world's Web server market and Microsoft holds approximately 30%. Web server market shares fluctuate from month-to-month, but the last 10-years' averages have been 50/30 for Apache and Microsoft (Netcraft, 2009).

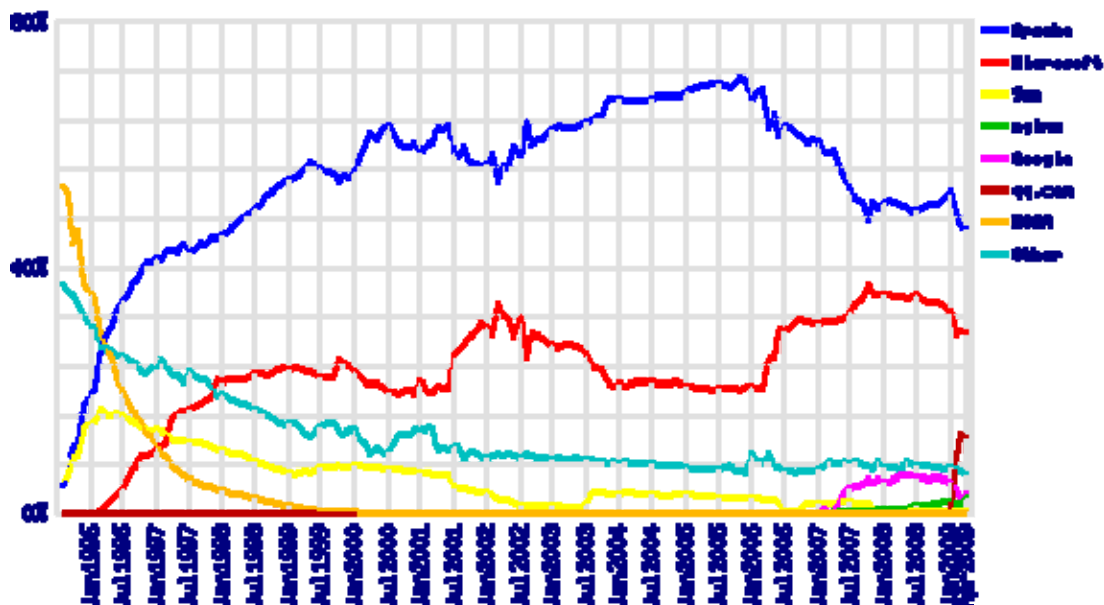


Figure 26. Web Servers' Worldwide Market Share (After Netcraft, 2009)

Microsoft's http server (Internet Information Server (IIS)) is relatively close in the market share dominance, and shows signs of taking over some of Apache's hare. This is largely due to the modern blog and chat tools built into the Microsoft server suite. Microsoft's general manager of servers, Bill Laing, believes Microsoft with its modern built-in add-ons has the potential to assume the lead in Web servers market share (Lai, 2008). Microsoft has relaxed some of its proprietary marketing models, and is adopting a

more open source approach to its products, including the integration of XML into their applications. Given this, Microsoft is another worldwide platform for EXI solutions to deploy.

Based on the dominance of Apache in the Web server market and its open business model, the OPENER-EXI solution, once stable, is being offered as an incubator contribution to be marketed to the Apache development group for implementation into their XML capabilities. Eventual acceptance and deployment of an EXI solution might bring EXI immediately into active and worldwide use. Additionally, with an EXI solution licensed with the ability to allow corporations to repackage the code without cost, it is likely to be adopted into more applications. If both Apache and Microsoft were to adopt EXI solutions, EXI would gain an 80% worldwide visibility among Web servers.

4. DoD EXI Implementations Considerations

DoD's Network-Centric vision equates to heavy leverage of the XML family of languages (ASD NII, 2006; DOD CIO IM, 2006; Langevin et al., 2008; Jacobs, 2008). EXI is being tested within DoD frameworks with success, and is likely to see adoption into the DoD information technology framework soon (MITRE, 2008).

a. EXI Must Become a Recognized Standard

Before DoD formally adopts any compression solution, EXI or other, it must be an official international standard. Without being a standard, support for continued development ranges from limited to nonexistent due to the narrow number of informed developers using the nonstandard solution. Nonstandard solutions, whether for cost or not, ultimately equate to proprietary solutions because only a few corporations will understand or will assume the cost and risk of maintaining a narrowly focused solution.

Nonstandard solutions only further propagate the existing stovepipe problems the DoD is trying to transition away. If a solution is not standardized, then other systems will not be able to interoperate. Each existing system would have to be

redesigned manually to enable them to collaborate with each new “Stovepipe,” which is expensive, inflexible to changes, and not likely to happen.

b. EXI Must Not be a DoD-Only Standardized Solution

Even if a solution passes various standards boards, and receives international standardization certification, that does not automatically equate to an acceptable and adoptable solution. In addition to being a recognized standard, a solution must also be adopted in many domains. Widespread adoption proves the solution is sound technically and is understood by many developers. Without wide adoption, a solution that has a standards certification becomes a pseudo nonstandard due to the reduced number of informed developers who invest time and effort into the discovery of the standard’s domain rules. With limited available developers and limited adaptation, the risk of loss of interoperability can repeat much like a nonstandard stovepipe solution.

An example of such an occurrences is the High Level Architecture (HLA), a standard simulation protocol (IEEE 1516) that was adopted by DoD, but not adopted by other simulation based corporations. Prior to the HLA standards intuitive, simulations from different developers could not work together which resulted many non-interoperable simulations stovepipe. The purpose of HLA was to enable arbitrary simulation developers to interconnect their simulation together seamlessly without requiring combined development effort. That is, a Boeing aircraft simulation could work with a Lockheed ship simulation out-of-the-box enabling DoD wide simulation of war-gaming or needs-gap-analysis.

DoD and a handful of other NATO militaries are the only entries that adopted the HLA standard. Because DoD and other militaries are not developers, they rely on corporate contractors to build their simulations. However, because only DoD and other militaries adopted the HLA and corporations did not, the cost of development sky rocketed due to lack of existing expertise in HLA within the simulation corporations. This lack of experts not only resulted in exceptional financial cost, it also resulted in increased interoperability cost because the few contractors that assumed the development of HLA simulations did not develop to a base HLA protocol, a Run-Time Infrastructure

(RTI). The RTI is the key to HLA interoperability, in that as long as all HLA simulations, called Federates, talk to the same RTI, they can interoperate. What actually happened is each contractor developed their simulations based on their own RTI, which disabled simulation interoperability between different developers. Figure 27 shows the RTI and Federate integration in general form.

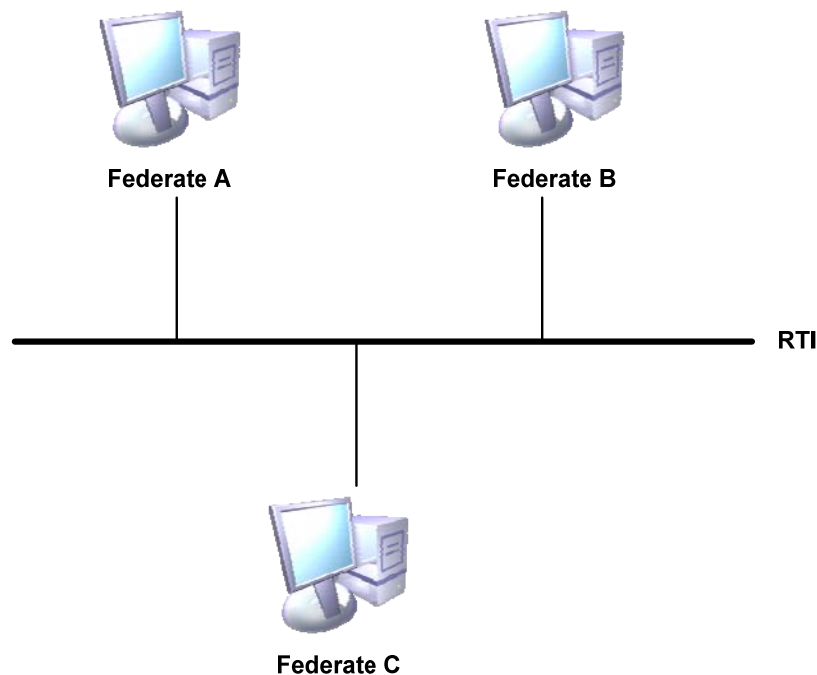


Figure 27. HLA General Architecture Overview Example

The result of HLA is a well-defined collection of standardized, but inoperable simulations. Ironically, counter to what the vision of HLA was trying to eliminate. HLA was for all intended purposes a failure as it ended up costing more and did not deliver interoperable simulations. Lack of interoperability requirements being set by the primary customer, the U.S. Department of Defense, is the primary root cause of this inevitable failure.

c. Possible Standalone Application

While the ideal implementation base for EXI is the Web server/browser integration, there is also merit for EXI as a standalone desktop application that would operate similar to that of WinZip or other desktop applications. However, by deploying EXI as a standalone application, several security and network administration concerns have to be considered in greater detail than if a part of a larger server suite.

(1) Security and Accreditation. DoD must approve all network software and devices before they can be installed on any DoD network architecture. EXI as a standalone application without a sponsoring activity, such as a Web server, as the backbone support forces would be forced to manage the entire security and accreditation process alone. While this is not an impossible task, the magnitude of the effort, confounded by lacking sponsor support, muscle, can potentially delay the deployment of EXI into the DoD networks for years.

A standalone DoD EXI implementation might require the EXI developer to conduct extensive and documented testing of the compatibility and security risk the EXI implementation poses to every network within DoD, or at least those networks that intend to implement EXI. This would delay the deployment of EXI, but would prove its security risk is low. However, if EXI were encapsulated in a sponsor's product, Web server, the sponsors, while simultaneously testing their array of applications, would also test EXI, and ultimately get EXI deployed faster and more reliably. Additionally, by simply having a trusted sponsor indirectly implies warranted trust and validation, which EXI as a standalone will not have until years of fleet usage.

(2) Network Administrator Workload Increase. As a standalone application, an EXI solution would have to be manually installed and maintained at each computer, not a single central point. This increases network administrator workload in terms of maintenance, installation, and most importantly, version control. As updated versions of EXI are released, those releases will have to be manually updated at each computer. Again, like the accreditation process just discussed, this is not an impossible task, but it does increase the likelihood of version control problems, which often equates

to a loss of interoperability due to codebase changes between versions. Further, as a standalone application, the probability of EXI being abandoned or inadequately supported increases.

d. EXI and DoD Integration Summary

Any DoD adopted solution must be interoperable with the existing network architecture, must be standardized by competent authority, and must be widely adopted in order to keep with the Network-Centric vision of a system-of-systems. If EXI does not receive W3C standardization endorsement and business-world acceptance, the DoD should not adopt EXI. If these prerequisites are satisfied, EXI is best suited for integrated within an existing sponsored program in order to simplify the integration and deployment process. Lastly, a HTTP server deployment of EXI can enable DoD the maximum leverage of EXI's potential as a majority of network XML traffic is HTTP based.

D. APACHE WEB SERVICES IMPLEMENTATION

The spirit of EXI is to enable deeper XML Web penetration to low-bandwidth locations and to small handheld and mobile devices, free of charge, and ready to deploy. The Apache Foundation is focused on providing a great Web server and other tools freely to the masses that also encourage open collaboration. There is a perfect alignment between EXI and Apache given the spirit of EXI and Apache's philosophy. As such, the OPENER-EXI implementation of EXI is initially being offered to Apache. Additionally, as already pointed out, Apache holds the dominant Web server market share around the world; embedding OPENER-EXI into the Apache foundation delivers an immediate worldwide impact.

1. How and Why Apache Is What It Is

The Apache development group started out as a handful of developers with the common interest of continued support and maintenance of the HTTPD Web server written by the National Center for Supercomputing Applications (NCSA) (Apache, n.d.).

Each member of the group collaboratively added code and shared ideas on how to make the server better. They got worldwide notice by allowing outsiders to listen in on the conversations through e-mail lists and to submit comments or code to better the project.

The origin of the name Apache is 3-fold (Apache, n.d.):

- For the Native American Indian tribe Apache that was known for their superior skills and inexhaustible endurance.
- A pun founded on the initial Web server project "a patchy Web server" because the Web server was constantly being patched by its members.
- As time ticked on, the developers started calling themselves the “Apache Group.”

In general, a group of people who needed a reliable Web server got together to build what the market did not provide. Because they were driven by necessity, innovation also occurred, which resulted in a great Web server that was not under any form of a proprietary license.

The magic that delivers the great code solutions to Apache is that many contributors are motivated by solution needs outside of corporate affiliation, and most importantly, only those who have shown to be good contributors are allowed to directly add to the codebase. The Apache Quality of Service (QoS) is what makes the Apache brand name the strong hold it is today, and this high level of quality has only been achieved by limiting who and what is added to the foundation: “We call this basic principle ‘meritocracy’: literally, government by merit” (Apache, n.d.). Further, since fast and powerful http serving is a widespread industry need, most companies can also align their goals and contributions with Apache to their own commercial benefit.

To this day, the Apache Software Foundation (ASF) is a non-profit organization focused on building great software that is freely distributable all users, public or private. The driving motivation is based on several simple but profound principles (Apache, n.d.):

- Provide a foundation for open, collaborative software development projects by supplying the hardware, communication, and business infrastructure.
- Create an independent legal entity to which companies and individuals can donate resources and be assured that those resources will be used for the public benefit.
- Provide a means for individual volunteers to be sheltered from legal suits directed at the Foundation's projects.
- Protect the “Apache” brand, as applied to its software products, from being abused by other organizations.

2. Establishing an Apache Project

Currently, Apache consist of 68 individual projects and countless numbers of subprojects, all listed in detail and freely downloadable at www.apache.org, the Apache website. To ensure that the same consistent quality of the original HTTP project is maintained in all new projects, Apache created a formal vetting process with three motivational points to validate all candidate projects (Apache, n.d.):

1. Developing according to the ASF’s philosophy and guidelines for collaborative development.
2. Ensure the legality of the code is for open distribution and transfer to the ASF under the Apache Software License (ASL).
3. Only products that meet the Apache's requirements are fully accepted into the ASF.

a. Vetting Process (Incubation) Overview

The process of bringing a candidate project into the ASF collection of projects begins in the Apache Incubation process (Apache, n.d.). This process starts with the candidate soliciting for a ASF sponsor and a proposal from the candidate to Apache describing the candidate project. If the ASF board accepts the candidate's proposal, the candidate enters a "podling" phase of development where it is refined and groomed into Apache pedigree. From the podling phase, a candidate may graduate into an Apache Top-Level-Project (TLP) or sub-project within an existing Apache Project. Figure 28 pictorially shows the path of progressing from candidate to ASF project member.



Figure 28. Apache Notional Project Adoption Vetting Flow (From Apache, n.d.)

The ASF bases all of its decisions democratically using a Project management Committee (PMC) majority vote with each TLP having its own PMC. This democratic process carries through for major decisions such as accepting new projects, to minor decisions such as deciding if a new software version should be released. The Incubator PMC is charged with the vetting of candidate projects through the ASF guidelines, vision, philosophy, and ultimately (if warranted) graduation into a TLP or subprojects.

Apache provides a mentoring process to assist candidates throughout the incubation process. During each phase of the Incubation process, a candidate has one or more mentors assigned that assist the candidate in the completion of phased tasks as well as the ASF transitions. Once accepted into the ASF, a former candidate becomes an eligible mentor to the next candidate project and also becomes its own PMC.

b. Becoming a Candidate Project (Pre-Podling)

The process from Candidate to Podling takes six steps, from finding a sponsor to acceptance into the Podling phase.

(1) Find A Sponsor. A candidate project must network with the existing TLP PMCs, the board of ASF, or the Incubator PMC in hopes that one of them will sponsor the candidate. The primary means of soliciting a sponsor is through mail-lists found on the Apache homepage. A sponsor is any PMC of the ASF, and has the duties of providing the initial review of the candidate before being presented to the Incubation PMC. This initial sponsor will become the candidate's mentor or will provide mentors once the candidate is voted into the incubation process.

(2) Build A Proposal. Once a candidate has a sponsor, it must then build its proposal to Apache. The proposal defines the project in terms of assumed benefits, known risks (dependencies, open source experience), Licensing, Cryptography, required resources (Subversion, mailing list), and a number of other aspects that define why the candidate is good, and how it is compliant with the Apache way of life and philosophy. There is no formal structure for the proposal, but the general guidelines and an example are available on the incubator proposal webpage (Apache, n.d.).

(3) Sponsor Vetting Proposal. The sponsor reviews and votes on the candidate's proposal. The sponsor applies its Apache experience to evaluate the candidate's preparedness for the incubation process and ultimately whether it is a good addition to the ASF.

(4) Sponsor Forwards Candidate Proposal to the Incubation PMC. If the sponsor approves the candidate's proposal by majority vote, the sponsor then forwards on behalf of the candidate to the Incubator PMC:

- Results of the sponsors Votes
- Candidates proposal
- Nominated mentors

(5) Incubation PMC Vetting. Once the Incubator PMC has the candidate's package of information, a 72-hour waiting period starts, after which, without a "hold" statement, automatically approves the candidate for the incubation process and entry into the Podling phase.

Any member of the incubator PMC can place a hold on the candidate. If a hold is placed within the 72-hours, a formal discussion and vote is conducted to make the final decision whether or not to accept the candidate.

(6) Podling Phase Approval. After being reviewed, and approved by the Sponsor and the Incubation PMC, the candidate, now podling, enters the Podling phase of maturing and refinement. Here the candidate is assigned its official mentor who will assist the candidate through the remaining steps of the process for integrating into the ASF.

c. The Podling Phase

The Podling phase is where the candidate project refines its software to meet the ASF quality requirements and "way of life": licensing, code release, distribution, and maintenance. Once a candidate project is in the Podling phase, the ASF is saying there is potential in the project, and hope to approve its acceptance into the ASF family of products once mature. The duration of the podling phases is not defined, since that is dependent on the speed of the podling project to transition itself into the Apache way of life.

(1) Podling Reviews. The incubator PMC performs regular reviews, quarterly or less, of the podling project's progress to determine if it needs to be terminated, continued, or graduation to a TLP. The self-assessment portion of the podling project's candidate proposal is the foundation of the first review. Ultimately, each review is a measure of the podling project's development according to the ASF philosophy and guidelines for collaborative development.

(2) Podling Project Pages and Mailing Lists. The incubator PMC, as well as all of Apache, uses project pages and archived mailing lists to track the status of all projects. To accommodate this, each podling project, through mentor assistance, will have to establish the following:

- The reporting schedule (milestone plans)
- The project status page (progress of the project)
 - Status of setup task
 - Exit criteria (graduation)
 - Status of exit criteria completion
- The mailing lists (collaborative networking)
- The repository space (code storage)

(3) Podling Check Points. The fundamental aspect of the podling phase is to weed out software that cannot meet the ASF philosophy and way-of-life standards. This is based on a number of common issues:

- Source Code licensing issues that will not enable open source.
- The code is overly dependent on external code that jeopardize the longevity of the podling.
- Not enough collaboration received or desired from the public.
- Fails to meet the ASF “look and feel” of how open source is supposed to work.

(4) Podling Termination and Termination Dispute. A podling can be terminated during any of the quarterly reviews if it fails to maintain schedule or meet the ASF way of life. However, if the podling or its mentor disagrees with the Incubator PMC findings for termination, they can dispute the claim. In receipt of a dispute, the Incubator PMC will review the podling project’s status. If the Incubator PMC finds the podling project worthy of continuation, it resumes its progress, but with added refinement tasks to insure improved progress.

d. The Podling Code Release Constraints

Part of the podling phase is the solicitation of support and interest in the podling project from the public; Apache brands great tools that have public value. To accomplish this, a podling will need to make numerous preliminary version releases to those interested in the project.

The ASF permits releases from their site, the podling repository at Apache, in the effort to further the podling codebase. They do however restrict any Apache brand from being associated to podling code, which is done to ensure the ASF name is not tarnished through immature podling code.

A podling must seek approval from the Incubator PMC for permission to release a code version, and do so only after all of the source-code is transferred to an ASF licensing (ASL). A podling must releases under the ASL, but cannot release under the name of Apache.

The process of approval of a release is conducted by majority vote, as is all other aspects of the ASF. First, a podling must have a majority vote from its mailing list to start a code release. Second, the podling sends a request to the Incubator PMC for code-release permission. Third, the Incubator PMC conducts a majority approval vote to determine code release approval.

With the Incubator PMC approval, the podling may then release their code, but with certain constraints:

- The word “incubating” must be in the filename.
- The release must contain the term “In Incubation” clearly visible in the main documentation or README file.
- Releases must be distributed through incubator approved channels:
 - <http://www.apache.org/dist/incubator/podling>
 - Maven repository

e. Graduation into the ASF

The end goal of the podling phase is graduation into the ASF as either a TLP or subproject. This determination will be made during the periodic Incubator PMC reviews based on the podling's ability to show:

- It is a worthy and healthy project
- It truly fits within the ASF framework
- It “gets” the Apache Way

These subjective criteria are achieved with the podling meeting the legal, community, alignment, synergy, and ASF infrastructure requirements as well as any additional requirements established by the Incubator PMC (Apache, n.d.). Does the code work, do people want the code, and is the codebase in-line with Apache's way of life?

The overall goal of Apache is to maintain the legacy of the namesake. Only codebases that are well tested, proven, desired, and mature may gain adoption into the Apache family. The candidate must transition its way of doing things into that of the way of Apache. However, through this burden of effort, a candidate reaps the benefits of Apache naming, process and sustainable success.

E. DOD SYSTEM ACCREDITATION PROCESS

The GIG policy memorandum No. 11-8450 states, all systems, sites, or applications must be certified as compliant with the GIG strategy, and must be verified that they do not inject any undue security risk or degrade any of the existing architecture (DOD CIO, 2001). This policy started in 1998, as DoD realized the added value a robust and secure information management system provided for current and future operations. The key motivation for this initial policy and all other policies that have spawned from it are the development of GIG policies and procedures that ensure a robust and effective network through: governance, resources, information assurance, information dissemination management, interoperability, network management, network operations, and computing.

Regardless of the path of integration (sponsored or standalone) that an EXI implementation follows into the GIG architecture, it will have to undergo the DoD Information Assurance Certification and Accreditation Process (DIACAP) (DOD CIO, 2007). However, depending on the path, the level of effort on the EXI developers will vary:

- If OPENER-EXI is a standalone, it will have to receive Certification & Accreditation (C&A).
- If OPENER-EXI can retain a sponsor then OPENER-EXI will only have to follow the Interface Certification Process (ICP).

Both processes are security focused designed to ensure any new system or application will not degrade the GIG or any of its subcomponents.

However, OPENER-EXI will likely be an embedded library of functions, making the product that uses OPENER-EXI burdened with the security accreditation process, not OPENER-EXI. The generalities of the security accreditation process are presented as guidelines and preparation in the event OPENER-EXI or other EXI solution is marketed as a pure standalone application.

1. Certification and Accreditation (C&A) Process

DIACAP defines the process and structure under which the Certification and Accreditation (C&A) process must follow as well as define the roles and responsibilities of those charged with the approval process in keeping with the GIG strategy. Specific task of information security (DOD CIO, 2007) and logistics (DOD CIO, 2008) are addressed to ensure adequate longevity is implemented into any adopted systems or application in terms of financial support and ability to meet the DoD missions.

a. DoD Information Assurance Certification and Accreditation Process (DIACAP)

The DIACAP process is documented by means of a System Security Authorization Agreement (SSAA). Every information system must be documented by an SSAA before the information system can request Authority To Operate (ATO), permission to connect, to operate or permission for installation, on the GIG. Depending upon the nature of the system/application, the SSAA can be of one of three types (DOD CIO, 2007):

- System—Major system application or a clearly defined independent system.
- Type—Common application or system that is distributed to a number of different locations.
- Site—Applications or systems at a specific, self-contained location.

OPENER-EXI will most likely be considered under the Type accreditation process.

b. Defense Information Systems Agency (DISA)

The agency in charge of ensuring all automated information technology systems and application are properly purchased, maintained and deliver the required security is the Defense Information Systems Agency (DISA) (DA&M, 2006). DISA manages the GIG and has the overall control of what systems, sites and applications that are connected. Their mission (DISA, n.d.):

Our goal at DISA is to ensure that our warfighters can plug into the network and access and share the information that they need, anytime, anywhere. We are dedicated to delivering the power of information as quickly as possible.

A warfighter's ability to leverage the right information at the right time is the difference between mission success and mission failure.

The warfighter's success is our mission.

DISA delegates ATO authority to its subagencies that have connections to the GIG: DON, DOA, CIA, HLS and any other agency that communicated over the GIG. Within each agency is a single entity titled as the Designated Approval Authority (DAA), charged with the review, certification, and accreditation of that agency's systems for the purpose of ATO approval or rejection (DoD CIO, 2007). The officer of the entity with ATO authority will be of significant pay grade, normally of the SES civilian pay scale. For example, within the United States Navy, the agency designated with ATO approval is Commander Naval Network Warfare Command (CNNWC) as declared by the Chief of Naval Operations, and the officer charged with ATO approval is of SES pay status (CNO, 2005). Each individual agency's DAA, may or may not, depending on the size and complexity of the agency, enlist subcomponents to carryout portions of the ATO process. Most commonly delegated is the Certification Agent (CA), which acts as a direct technical advisor to system developers in an effort to assist and perform quality assurance of the SSAA submissions before being sent to the DAA. The CA does not have approval authority, but their inputs are of significant value to the DAA along with the SSAA in the ATO determination.

c. System Security Authorization Agreement (SSAA)

The SSAA is the fundamental document that defines the system used in the making of the ATO determination for each system, new or old (DOD CIO, 2007). Once ATO has been issued, it is good for up to 5-years, assuming no changes are made to the system outside of what is declared in the SSAA. After 5-years the system must repeat the DIACAP process and generate a revised SSAA.

The SSAA document generation is extensive and often takes well into a year, if not years to develop. In the interim during SSAA development, the DAA can issue an Interim Authority to Operate (IATO), which grants temporary, but restricted GIG access to the developing system. However, this temporary IATO will only be issued after a significant portion of the SSAA documentation has been generated (DOD CIO, 2007).

The concept of the IATO is to get tools into productive use as soon as possible, but only after a reasonable level of security is proven. Generally, the only portions of the SSAA that can remain for an IATO are the administrative sections such as training and recovery policies. The core of the SSAA must have already been developed and approved by the CA: security policies and configuration. A high-level overview of the flow for a new system to receive ATO is shown in Figure 29.

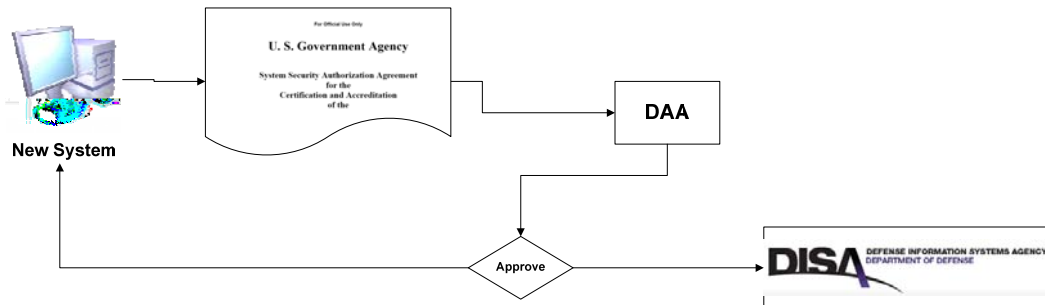


Figure 29. High Level Overview of the DIACAP Process

The DoD Information Technology Security Certification and Accreditation Process (DIACAP) instruction DODI 8510.01 contains the overall flow and structure of the SSAA document. Templates and outlines can be obtained from DISA by request. A good starting point for any SSAA is to borrow from an existing system since the overall chapter layout is the same for all systems.

2. The Interface Certification Process (ICP)

A less formal and less intense process of introducing a new application onto the GIG is the Interface Certification Process (ICP). This process builds off an existing system, a Program of Record (POR) already in service that holds an ATO. The new system or application submits to the POR a request asking them to amend their system with the new application or system. The General overview of the ICP starts with software development and Interface Change Request (ICR), and if approved, goes to Engineering Change Plan (ECP) for implementation, as depicted in Figure 30.

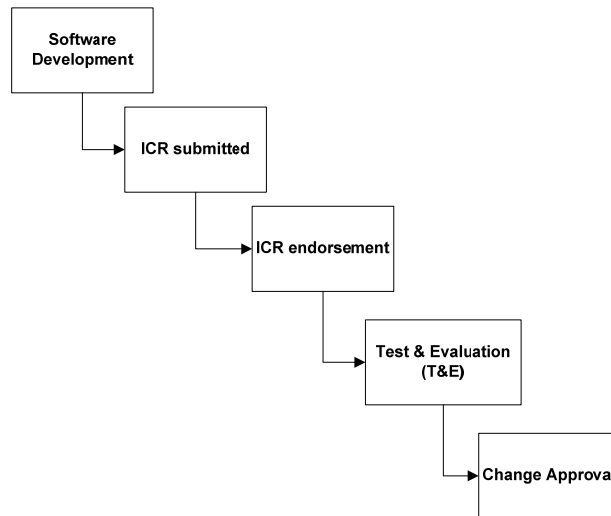


Figure 30. Notional ICR Progression Events Waterfall Chart

All POR have their own ICP process, though all follow common DoD guidelines to ensure new software is assessed, certified, and interoperable with the existing software packages (Figure 31). The controlling DAA of the sponsor POR designates a Configuration Manager (CM) who is charged with the processing and evaluation of all ICR for its POR area of responsibility. The designation of a POR CM reduces the time delay from solution development to integration, and it enables Type Commanders (TYCOMS) with a direct means to control their networks to meet mission needs. The CM objectives are to test and evaluate all ICR and provide reports of the same:

- Interoperability
 - Ports and Protocol management
 - Network utilization / Bandwidth control
- Installation
 - System Configuration
 - Uninstall procedures
 - Funding
- Security
 - Vulnerabilities
 - Update and patch management

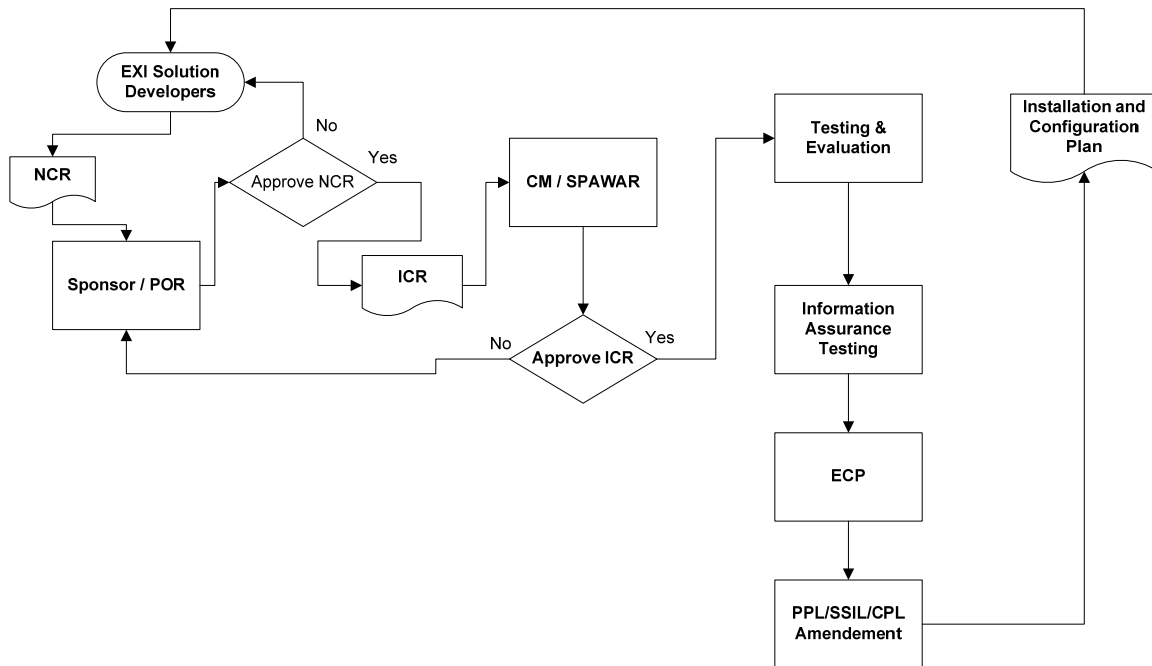


Figure 31. DoD Afloat Change Request Flowchart Example

a. U.S. Navy Afloat ICP Example

A case example of the ICP process is the U.S. Navy (USN) Afloat network POR. For the USN Afloat POR, Commander Navy Network Warfare Command (CNNWC) is assigned as the DAA, and has designated Space and Naval Warfare Systems Center Pacific (SPAWAR) as the CM to evaluate all ICR for the afloat networks. All directives, submission and tracking processes for the USN Afloat networks are maintained at SPAWAR's Web site <https://navalnetworks.spawar.navy.mil/>. This site is the official source for afloat networks, specifically listing what can and cannot be on afloat networks. The site also provides an interface for submitting change request to the approved list of items. SPAWAR maintains three lists of approved, installable items that have been assessed not to interfere with the existing afloat architecture. Only items found on these lists may be installed on an afloat network:

- Preferred Product List (PPL): What software is deemed safe and approved for installation.

- System/Subsystem Interface List (SSIL): What systems are approved for integration into an afloat network.
- Certified Parts List (CPL): What network components (routers, switches, printer, etc.) are approved for integration into an afloat network.

b. U.S. Navy Afloat ICR Submission Process

The initiation of a change to a network (such as adding OPENER-EXI as approved software) begins with a Network Change Request (NCR) from the developer or fleet user to the CM. The NCR requester does not submit an actual ICR, only a POR can submit an ICR. This affords the POR control over their programs by allowing them veto authority before an ICR reaches the DAA.

A NCR is initiated on the SPAWAR site requesting an update to the one or more of the list of approved items and directed to the sponsor POR. The NCR defines the item in terms of its applicability and need along with initial Test and Evaluation (T&E) reports. The exact submission of the NCR is relative to the type of change. If the NCR is simply asking that an updated version of an approved application with an established process be authorized for use, the NCR will be very simple. If the NCR is asking for permission to add something without a history, such as OPENER-EXI, the submission will require much more details.

c. U.S. Navy Afloat ICR Endorsement

The sponsoring POR receives the NCR and evaluates the validity and accuracy of the request. After review and approval of the NCR the POR creates an ICR in the name of the NCR submitter. POR are the only ones that can submit an ICR because they own the system. Within the ICR, the change requests must define the system and the application, list previous approved updates, and a proposal for ECP funding plan as applicable.

Beyond the technical aspects of a new item, the change must be supportable over the expected life of the item; technical support factors such as training have to be planned and accounted. The POR has the onerous obligations, based on initial NCR, of mapping the life of the new item: training, funding, installation and life cycle support. This is documented in the ICR and forwarded to the CM for review, T&E, and ultimate ATO on the DoD network under the supervision of the POR if approved.

d. U.S. Navy Afloat CM Assumes the ICR for Test and Evaluation (T&E)

The CM receives the full ICR from the POR and validates the request. If the CM finds the ICR to be of value, the CM then forwards the request to the T&E phase. The purpose of the T&E is to verify and validate the risk that the new item poses to the existing architecture of both the POR and DoD in general. The exact tests conducted vary depending on the scope of the change item. OPENER-EXI for example, will likely undergo network vulnerability scanning and interoperability testing with the existing USN afloat architecture components. Since OPENER-EXI is scoped for the Web server environment, tests will be conducted to ensure the existing server software is not impacted by OPENER-EXI. For example, if an application must use GZip as its transmission compression, test will be run to verify that OPENER-EXI does not override GZip, and that those application that are EXI compliant can use the implemented OPENER-EXI version.

Successful exiting the CM T&E occurs after extensive testing shows the ICR item poses no degradation to existing architecture, and moreover, that the requested change adds value that did not previously exist.

e. U.S. Navy Afloat ECP and Installation

With successful validation of the ICR by the CM, an ECP is generated that list the funding, installation plan, and training for all commands that will receive the ECP. For OPENER-EXI, there is likely no funding required, only authorization to add. Additionally, the corresponding list, PPL, SSIL or CPL, will be updated to reflect the

approval enabling the installation on any of the POR's networks. OPENER-EXI would be reflected on the PPL list given it is an application.

Ultimately, the ECP is the official funding and approval to add an item (OPENER-EXI) to the DoD network.

F. CHAPTER CONCLUSION

OPENER-EXI must be licensed as a non-viral Open source Software to ensure successful adoption. The best EXI deployment platform is the http server/browser environment. A good pairing of both open source and Web is the Apache Software Foundation as sponsor of initial implementation and deployment efforts.

DoD should only leverage standardized tools and solutions, including EXI, that are also widely used throughout the Information Technology world. Without adhering to these points, the DoD risks ending up with a modern stovepipe that is unable to interoperate with existing and future systems.

An EXI application will benefit from integration into an existing POR sponsorship. Without a POR sponsor, EXI is subject to pursuing the DIACAP alone, a process that is likely to prove too costly and time consuming to pursue for a standalone application.

G. CHAPTER SUMMARY

This chapter discusses the administrative considerations of an EXI solution in terms of licensing, deployment considerations, and DoD specific constraints. The NPS EXI implementation, OPENER-EXI, is used as the example case study.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. EXISTRUCTURE AND OPENER-EXI IMPLEMENTATION

A. INTRODUCTION

This chapter describes the W3C EXI specification and how OPENER-EXI implements the specification. The chapter starts by describing the EXI header format followed by the EXI techniques used to parse XML documents into compact and efficient EXI streams. The test case XML documents used to test the OPENER-EXI implementation are listed with a description of the subset of the EXI specification in each exercise. The chapter concludes by summarizing the software engineering practices used in the development of OPENER-EXI.

B. PREAMBLE

1. Source of Reference

The W3C's general development guidance for the EXI specification follows five principles: the format has to be general, minimal, efficient, flexible, and interoperable. The preliminary W3C specification for EXI is published in their *Efficient XML Interchange (EXI) Format 1.0* document (W3C, 2008) and introduced in general form within their *Efficient XML Interchange (EXI) Primer* (W3C, 2007) document. Without explicit statement otherwise, these documents provide all the resources used for this chapter to define the specification and in the development of the OPENER-EXI implementation.

2. Chapter Goals

The intended goal of this chapter is to develop a “one-stop shopping location” that consolidates all the EXI references and key points into one location, and do so in an easy to follow format with understandable terms. While the entire EXI specification, as well as its evolutionary development is well documented and freely available, locating the references and their understanding is not likely to be immediately, which is the

motivation for this chapter: to simplify and consolidate the resources and specification. Given this, the only original input contained in this chapter are the OPENER-EXI code samples, simplifications of tough concepts, subprocedure outputs, and the organizations of the presentation of concepts. Always refer back to the specification for the most current revisions and changes to the standard.

3. EXI Introduction

When refereeing to XML it is common to use the term “document.” but in EXI, the term is “stream.” EXI is a steam of events, and not a traditional document. This detail is minor to the understanding or appreciation of EXI, but is essential for the understanding of the lingo. Essentially XML and EXI are both just files, but each has its own unique labels.

Each EXI stream consists of two basic parts (Table 22): a header and a body. The header, section C of this chapter, instructs an EXI processor about the stream, that is, how to encode and decode. The body, section D of this chapter, consist of the sequence of events describing the original XML document. The header and body are the basic building blocks of every EXI stream.

EXI Header	EXI Body [stream of events]
-------------------	---------------------------------------

Table 22. Basic EXI Stream Structure

C. HEADER

Like most other file formats, EXI employs a header at the beginning of each stream in the format of Table 23. The absolute smallest EXI header will consist of 2 bytes 1001 0000 (hex 0x90 or decimal 144) using all the default settings, but can grow to an unlimited size because there is no technical bounds on the format version field size.

[cookie] + Distinguishing Bits	Options Presence Bit	Version	[EXI Options]	Padding bits
---	---------------------------------	----------------	----------------------	---------------------

Table 23. EXI Header Format

1. Distinguishing Bits and Optional Cookie (Header Part 1 of 5)

Every EXI document starts with two distinguishing bits 1 and 0, and in that order. These two bits are all that is needed to uniquely distinguish EXI from other text-based XML streams. An EXI header may optionally contain, and before the two distinguishing bits, an EXI cookie consisting of a four byte field ‘\$’, ‘E’, ‘X’ and ‘I’. The cookie is optional, but recommended in the specification whenever a “...more solid content-based datatype identification is desired than what is provided by the Distinguishing bits.” This longer identifier may be desirable if EXI is exchanged with file formats other than XML. The cookie makes EXI differentiation from all other file format types easier. Table 24 and Table 25 provide examples of the distinguishing formats.

Note, items contained within single quotes are ASCII 8 bit characters and those without single quotes are single bits.

1	0
---	---

Table 24. Minimum Bits to Distinguish an EXI Stream from Other Text-Based XML Streams

‘\$’	‘E’	‘X’	‘I’	1	0
------	-----	-----	-----	---	---

Table 25. EXI Stream Distinguishing Bits with Optional 4-Byte Cookie

Any deviation from these EXI stream identification formats, the EXI processor is expected to reject the input EXI stream.

2. Options Presence Bit (Header Part 2 of 5)

Flags to the EXI processor whether or not any EXI options were set during the encoding process needing to be handled for the decoding process of EXI to XML. Because the EXI specification defines a default state for the creation of an EXI streams, only deviations from the defaults need to be indicated. As will be addressed in header part 4 of 5, the listing of options used is placed within the “EXI options” part of the header. Since the EXI stream is either build based on default options or incorporates additional options, a single bit is all that is needed to represents these two states; a 1 for additional options used and 0 for defaults only.

3. Format Version (Header Part 3 of 5)

As EXI evolves, it is likely to undergo a number of revisions, and by design, possibly a number of domain-specific revisions. Therefore, to prepare for the longevity of EXI, the version indicates to the processor which EXI implementation the input EXI stream was encoded. The version consists of an EXI unsigned header integer, which is a series of one or more 4-bit integers prefixed with a version distinguishing bit. Note the EXI header integer is not an IEEE integer or any of the various other EXI stream integer types discussed later in this chapter.

The first left most bit holds significance, indicating whether or not the encoding was done with a preview or final version of the EXI format. A 0 indicates a final version and a 1 indicates a preview version. A final version corresponds to an approved W3C standard version of the EXI specification and a preview is anything else. OPENER_EXI is a preview version and such has the flag set to 1. A note in the specification requires processors that implement a final version of EXI to process all EXI streams that have the first bit of the format version set to 0, and it is at the option of the processor to process preview versions.

After the version flag, the version of the implementation is represented as a series of one or more 4-bit unsigned integers. The version number is achieved by summing the series of 4-bit integers, from left to right, and is terminated at the first 4-bit integer that is less than (1111), which is EXI version 15 +.

Based on 4-bits, each unsigned header integer can take on values in the range of one to fifteen [1, 15]. It is important to note that a version of 0 is not included in the range of values, the four bit header integer 0000 equates to 1 and not 0 as would normally be expected with binary values. A simple way to solve for the decimal value of each EXI 4-bit header integer is to add 1 to the value normally expect if the 4-bit pattern was a binary number. Table 26 demonstrates some instances of the 4-bit pattern for the header integer without the version flag.

4-Bit Integer (EXI)	Decimal Value for EXI Version
0000	1
0001	2
0010 – 1101	3 - 14
1110	15 and terminal
1111	15 + next integer

Table 26. EXI Header 4-Bit Version Unsigned Integer Examples without Prefixed Version Flag

For example, the default EXI header with 1001 0000 states from the left:

- 10–EXI distinguishing bits
- 0–no options present
- 1–preview version of the EXI Specification
- 0000–first version

The value of the bits of each 4-bit integer is read from right to left, though series of 4-bit integer values are summed from the left to the right. This allows for an

unbounded but compact version field. Table 27 describes the decoding pseudo code for EXI version number, and Table 28 list some examples of 4-bit header integers.

```

1.  Set version = 0
2.  Set Current4 = Read the next 4 bits
3.  Set version = version + current4 + 1
4.  if Current4 = 15
    version -= 1 // would be 16 otherwise
    goto step 2
    else return version

```

Table 27. EXI Header Version Number Pseudo Code

VERSION FIELD	DESCRIPTION
1 0000	Preview version 1
0 0000	Final version 1
0 0001	Final version 2
0 0010 – 0 1101	Final versions 3 - 14
0 1110	Final version 15
1 1111 0000	Preview version 16
0 1111 0001	Final version 17

Table 28. EXI Header Version Field Examples

4. Options (Header Part 4 of 5)

EXI provides several encoding options that enable EXI streams to be created with lossless or lossy considerations; lossy encoding being unable to recreate the exact input XML document at decode, and lossless being able to recreate an exact replica of the input XML document. Table 29 lists the available EXI options, their default settings and a short description of the options impact on an EXI stream. These options when exercised enable a higher level of compactness; although such compactness can come at an efficiency cost depending on the domain of interest. An introductory example of when

these options must be carefully exercised is when dealing with XML security and XML encryption, both discussed later in this chapter.

EXI Options	Description	Default Value
Alignment	Alignment of event codes and content items	Bit-packed
Compression	EXI compression is used	False
Strict	Strict interpretation of schema is used	False
Fragment	Body is a fragment and not a full XML document	False
Preserve	Specifies whether or not to preserve	All false
Self Contained	Enables self contained elements	False
Schema ID	Schema ID used to encode the EXI	None
DatatypeRepresentationMap	Datatype used to encode values in EXI body	None
Block Size	The blocking size for EXI compression	1,000,000
Value Max Length	Largest string that can be added to the string table	Unbounded
Value Partition Capacity	Maximum capacity of the VALUES portion in the string tables	Unbounded
User Defined	User defined options	none

Table 29. EXI Header Options and Default Values (From W3C, 2008)

When options are employed, they are defined within an EXI options XML document based on the options schema listed in Table 30. This EXI options XML document is then encoded into the EXI options header field using the default EXI options settings except the option of byte-alignment is used instead of the default bit.

<pre> <xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema targetNamespace=http://www.w3.org/2007/07/exi elementFormDefault="qualified"> <xsd:element name="header"> <xsd:complexType> <xsd:sequence> <xsd:element name="lesscommon" minOccurs="0"> <xsd:complexType> <xsd:sequence> <xsd:element name="uncommon" minOccurs="0"> <xsd:complexType> <xsd:sequence> <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/> <xsd:element name="alignment" minOccurs="0"> <xsd:complexType> <xsd:choice> <xsd:element name="byte"> <xsd:complexType/> </xsd:element> <xsd:element name="pre-compress"> <xsd:complexType/> </xsd:element> </xsd:choice> </xsd:complexType> </xsd:element> <!-- alignment --> <xsd:element name="selfContained" minOccurs="0"> <xsd:complexType/> </xsd:element> <!-- selfContained --> <xsd:element name="valueMaxLength" minOccurs="0"> <xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"/> </xsd:simpleType> </xsd:element> <!-- valueMaxLength --> <xsd:element name="valuePartitionCapacity" minOccurs="0"> <xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"/> </xsd:simpleType> </xsd:element> <!-- valuePartitionCapacity --> <xsd:element name="datatypeRepresentationMap" minOccurs="0" maxOccurs="unbounded"> <xsd:complexType> <xsd:sequence> <xsd:any namespace="##other"/> <!-- schema datatype --> <xsd:any namespace="##other"/> <!-- datatype representation --> </xsd:sequence> </xsd:complexType> </xsd:element> <!-- datatypeRepresentntationMap --> </xsd:sequence> </xsd:complexType> </xsd:element> <!-- uncommon --> </pre>	<pre> <xsd:element name="preserve" minOccurs="0"> <xsd:complexType> <xsd:sequence> <xsd:element name="dtd" minOccurs="0"> <xsd:complexType/> </xsd:element> <xsd:element name="prefixes" minOccurs="0"> <xsd:complexType/> </xsd:element> <xsd:element name="lexicalValues" minOccurs="0"> <xsd:complexType/> </xsd:element> <xsd:element name="comments" minOccurs="0"> <xsd:complexType/> </xsd:element> <xsd:element name="pis" minOccurs="0"> <xsd:complexType/> </xsd:element> </xsd:sequence> </xsd:complexType> </xsd:element> <!-- preserve --> <xsd:element name="blockSize" minOccurs="0"> <xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"/> </xsd:simpleType> </xsd:element> <!-- block size --> </xsd:sequence> <!-- less common sequence --> </xsd:complexType> </xsd:element> <!-- less common --> <xsd:element name="common" minOccurs="0"> <xsd:complexType> <xsd:sequence> <xsd:element name="compression" minOccurs="0"> <xsd:complexType/> </xsd:element> <xsd:element name="fragment" minOccurs="0"> <xsd:complexType/> </xsd:element> <xsd:element name="schemald" minOccurs="0" nillable="true"> <xsd:simpleType> <xsd:restriction base="xsd:string"/> </xsd:simpleType> </xsd:element> </xsd:sequence> </xsd:complexType> </xsd:element> <!-- common --> <xsd:element name="strict" minOccurs="0"> <xsd:complexType/> </xsd:element> </xsd:sequence> </xsd:complexType> </xsd:element> </xsd:schema> </pre>
---	---

Table 30. Options Schema for EXI Header Options XML Generation (From W3C, 2008)

There is no specified method defining how to handle options when options are not listed within the EXI header but were used in the encoding. That is, the EXI stream's header options present bit is 0 and the options field is empty, but options are present in the encoding of XML to EXI. The specification does note that the use of anything other than the default values would likely be in a control system where specific preexisting knowledge about the EXI format is known. The likely best choice outside of preexisting domain knowledge would be to assume the default options, as indicated in the header. Potentially this loophole might be used to hide information within an EXI stream if the EXI processor does not note it.

a. Alignment Options

As the title of the option suggests, this controls the alignment of the body of the EXI stream. The available encodings are: bit-packed, byte-aligned, pre-compressed, which is blocked and channelized but not compressed, and compressed. The default alignment is bit-packed. Byte-aligned is required for many domain-cases, such as digital signatures and encryption, and is also a good option to use when troubleshooting given the results are in ASCII aligned format, readable in notepad and other simple text-editors.

b. Strict Option

The Strict option is normally set to false, but when true, this option prunes (discards or ignores) namespace, comment, processing instructions and self-contained events from the input XML document. The strict option relies strictly on the supplied schema for the EXI stream structure, only using the XML document for values: attribute values and element content. At decode, through the same scheme used at encode, all pruned events, other than comments, can be reconstructed based on the schema. When exercised this option provides for a more compact stream.

Caveats to this option are it can only be used if a schema is provided, and any deviation within the input XML document from the provided schema will result in a fatal EXI processing error. Strict, as its name implies, enforces strict schema compliance for input XML documents.

c. *Fragment Option*

Normally false, this option indicates whether the stream is a document or a fragment. A fragment is a sequence of well-formed XML elements or processing instructions, that although appears like a XML document, are not standalone valid XML. For example, fragments do not have an XML header `<?xml version="1.0" encoding="UTF-8"?>` which a standalone XML document contains, but a fragment may contain elements and attributes.

d. *Preserve Options*

Normally false, EXI enables the ability to prune certain events from an XML document that do not impact the content of the document for some applications. Often these event items can be removed without impacting any aspect of the XML file, and for compactness considerations, can be stripped from the EXI stream while retaining the spirit of the original input XML document. Table 31 describes the pruneable event options. However, none of the preserve options can be exercised if the strict option is used.

Fidelity Option	Effect
Preserve.comments	Retains any XML comments within the document
Preserve.pis	Retains any processing instructions within the document
Preserve.dtd	Retains any DTD within the document
Preserve.prefixes	Retains any namespace prefixes within the document
Preserve.lexicalValues	Lexical form of elements and attribute values preserved

Table 31. EXI Fidelity Options: Event Preservation Options (From W3C, 2008)

Any pruned events during the encoding of an EXI stream from an XML document are lost, and cannot be reconstructed directly from the EXI stream when decoding, and so such encoding is lossy. These pruning options can normally be employed without risk, but domain-case specifics must be taken into consideration before their employment is used.

e. Self-contained Option

Normally false, this option enables faster indexing through elements that are read independently of the rest of the EXI body. The self-contained cannot be used if the compression or the pre compression alignments are used; it is only applicable to bit and byte alignments.

f. Schema ID Option

Normally omitted, this option identifies the schema used to encode the stream. The format of this field is not defined and is left to the implementers and users. This field might be a URI or other indicator that enables identification and retrieval of the schema of interest. This allows for domain-specific schema-aware document encoding algorithms so that unique domain-cases can leverage their architecture to achieve the maximum compactness possible. An example of this approach might be a production system that uses only one schema, and always uses that schema. Such a production system could forgo the identification of the schema for a hard-coded schema design directly within the EXI processor, by knowing that all EXI streams it will encounter are expected to conform to the one known schema.

g. Datatype Representation Map Option

Normally omitted, like the schema ID field, the Datatype Representation Map field uniquely identifies a list or map of datatypes used in the encoding of this stream. Data maps are covered in more detail later in this chapter within the datatype representation map section.

h. Block Size Option

This defines the data block size for compression windows with default size of 1,000,000. The EXI compression technique works on blocks of data, and this field defines the maximum size of each block. The EXI compression methodology is covered with greater detail later in this chapter within the compression section.

i. Value Max Length Option

Normally unbounded, this indicates the largest string that can be added to the “value” portion of the string tables. Note that string tables are essential components of EXI, and are covered in detail later in this chapter. If the character count of a string is larger than this maximum length, the string is not added to the string table. Instead, it is written directly to the EXI stream as an ASCII string literal. This ensures one-time long string values do not clog up a string table with unique occurrences, such as a multi-sentence paragraph within an element. The reasoning for this approach is that one-time paragraphs will not likely be repeated exactly within an XML document. Adding them to the string tables and creating indexes cost more in terms of compactness than writing the original paragraph directly to the EXI stream.

j. Value Partition Capacity Option

Normally unbounded, this option specifies the maximum number of strings allowed for the global and value string tables. Note, string tables are essential components of EXI, and are covered in detail later in this chapter. Each new string encountered within the XML document is added to the string tables only if there are less than Partition Capacity items currently within the tables. In either case, the string is written to the EXI stream as ASCII for the first occurrence.

k. User-Defined Option

This field is undefined for future or domain-case specific needs.

5. Padding Bits (Header Part 5 of 5)

The EXI header, unlike the EXI body, must conform to byte-alignment. The reason for this alignment requirement is there are no events that define the header so header parsing requires byte alignment in order to determine header-section boundaries. Therefore, as the previous header sections are created, it is possible that the header will not end on a byte boundary. The padding bits insert enough bits to terminate the header on a byte boundary.

The padding bit field will either not exist if the header naturally falls on a byte boundary or it will at most contain 7 bits. There is no specified requirement for the padding bits so they can be either 1's or 0's, although 0 implies a more intuitive padding indicator.

Referring back to the absolute smallest header of 1 byte 1001 0000, it can be determined that this header consist of distinguishing bits (10), the absence of options (0), the first preview version number (1 0000), and consist of no padding bits as this minimum header conforms naturally to a byte boundary.

6. Graphic User Interface (GUI) Tool for EXI Options

Figure 32 is a graphic user interface (GUI) built for the OPENER-EXI implementation that captures and exercises all EXI options and settings, including the interactions of options.

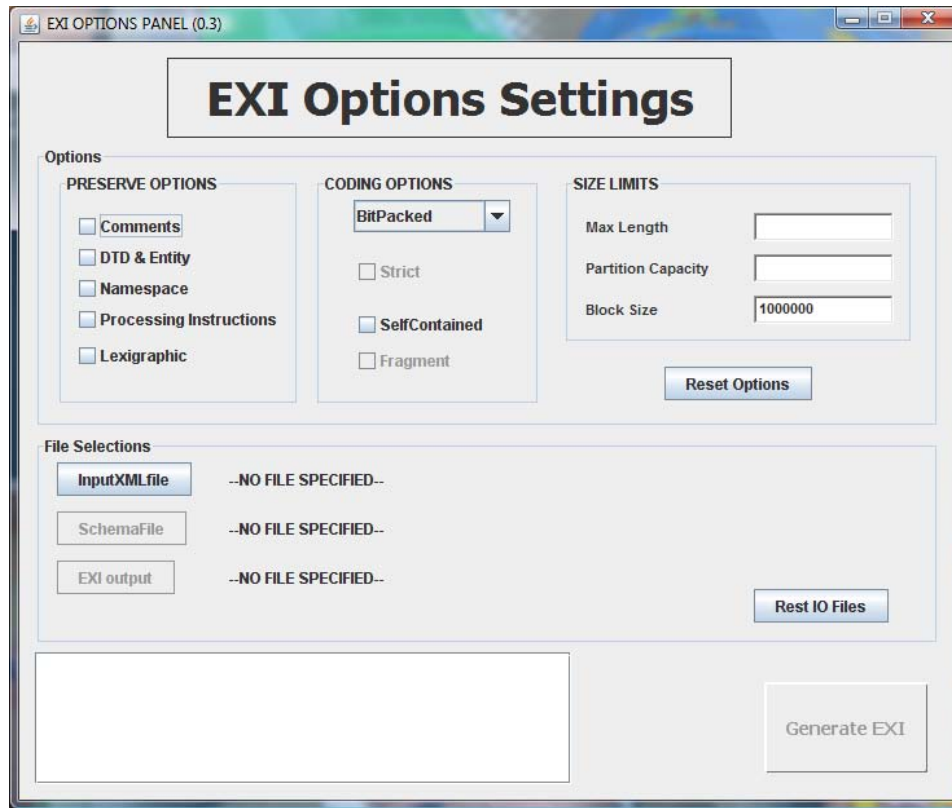


Figure 32. OPENER-EXI's Graphic User Interface Capturing All EXI Options and Settings

D. EXI BODY

EXI's ability to achieve superior compression and efficiency is in its XML-specific structure awareness, which is not utilized by other compression techniques such as GZip. By taking advantage of the XML-specific redundant data structuring, EXI delivers excellent compression. When a supporting schema for the input XML document is leveraged, superior XML compression and efficiency is realized using EXI's binary datatype bindings that exceeds all other compression techniques for XML documents.

EXI builds the XML document structure in memory by means of learning grammars that consist of XML events. These events represent the XML content and are identified with a compact code of a sequence of 1 to 3 non-negative integers called parts.

String tables are a technique common to most text-based compression techniques, and are also used in EXI. String table lookups map redundant string values to compact identifiers that uniquely identify all string values.

By XML schemas, the EXI body can achieve higher levels of both compression and processing efficiency beyond redundant string elimination. Higher compression is achievable by pre populating the string tables with entries found within the schema, as well as pre mapping the schema structure of the XML document into memory before processing the input XML document. Efficiencies are gained from the schema by allowing EXI to encode XML content in binary format based on the datatypes defined within the schema for the XML content.

Additional compression can be achieved by applying a second compression technique on the EXI body during the EXI encoding of values.

Overall, the body of an EXI document is a series of XML events following the general procedural flow listed in Table 32.

- | |
|--|
| <ol style="list-style-type: none">1. Get the next event from the XML document2. If a fidelity option says not to process, go back to step 13. Use the current grammars to determine the event's code4. Determine if the events string content is a repeat or new value based on the string tables, and the set the content value5. Write the event code followed by the event content to the EXI stream6. Update the grammar productions with the event7. If not end of document event, go to step 1 |
|--|

Table 32. EXI Stream Encoding Pseudo Algorithm

1. String Table

String tables are the backbone of the compact identifier assignment process used by EXI. Each XML content item, element content, attribute value or namespace events is remembered in a string table when it was first seen. If seen again, this repeated occurrence is represented as a compact identifier, instead of the string literal, which is an index into a string table that contains the first occurrence of the same string.

Even if an XML document contains a large number of unique string values, compactness savings are still realized by not encoding the XML formatting characters into the EXI stream. In the worst case, every string in the XML document is unique, a file size savings is still achieved by not writing the “<”, “>”, “</” XML formatting characters because the EXI events are encoded with bit identifiers instead of byte size characters.

a. Building the String Tables

String tables are XML namespace aware, and can be thought of as a list of lists of strings, with a namespace being the root of each list. For each namespace, several other tables, also called partitions, are created and associated to the root namespace as shown in Figure 33: prefix, local-name, local values and global tables. It should be noted that a string table itself does not get transmitted with the EXI stream, and that a string table cannot be used across multiple streams. Every EXI encoding creates its own string tables particular to that encoding.

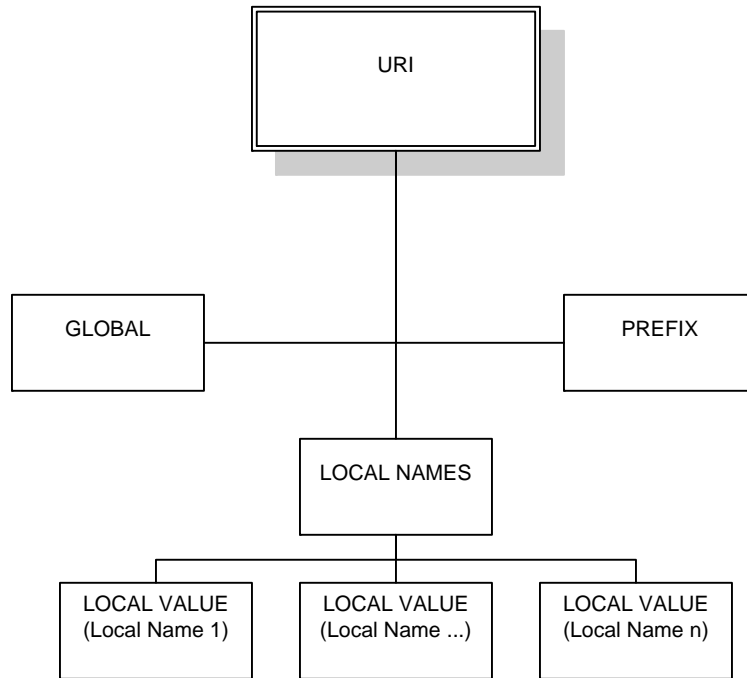


Figure 33. Basic EXI Namespace Driven String Table Design

When a schema is provided, the string tables are built before processing the XML document based on the schema's defined attributes and elements names, along with EXI default values. If no schema is provided, the EXI technique builds the string tables as it encounters strings, but still contains a default structure that is populated with EXI default values.

When strings values are encountered while processing an XML document, their first occurrence is added to the global and local value table of the firing event's associated namespace local name table portion, and then written to EXI stream as raw ASCII text. Subsequent occurrences of the same string value are not added to the string table, and instead of raw ASCII output, the index to the sting within the string table is written to the EXI stream.

b. Data-structure of the String Table

The OPENER-EXI data-structure data structure of choice that best encapsulates string tables is a HashMap. The HashMap structure takes two parameters, the value of an item and the item's key. In EXI string table terms, the value is the string literal and the key is the string's compact EXI string table identifier or integer index. A HashMap is essentially a table-lookup data structures that allows for non-continuous keys and values to be efficiently stored and searched, and can also dynamically grow as the number of values grow. HashMaps have another nice feature well suited for EXI: they only add unique values to their structure. That is, a HashMap will not allow duplicate values to be added to the structure. This is convenient in that it ensures no redundant string values are recorded inherently by its design.

Each global, local value and name partition of a Namespace of tables contains two maps, one for string to integer and another from integer to string; each with its corresponding getter/setter methods to set and get string values by index or else index to string.

```
private HashMap<String, Integer> stringToCompactIdentifier;  
private HashMap<Integer, String> compactIdentifierToString;
```

Using these two HashMap structures enables queries for both string and index values. When adding strings, the <String, Integer> is used, and when getting a string at decode the <Integer, String> map is used. The key to this technique is the synchronous upkeep of both structures. This is best accomplish by wrapping both of the structures within a *StringTable* class that performs, by means of function calls, the adding and getting of string values while it keeps track of index house keeping

The root for each string table set is a namespace, which itself is a string table. Using the same HashMap structure just defined, but with different HashMap keys and values, both the namespace string table and prefix string table are made.

```
private HashMap<String, Tables> URITables  
private HashMap<String, String> prefixTables
```

The `HashMap<String, Tables> URITables` structure contains each namespaces tables, defined next, as the value and the URI's string value of the namespace as the key. The `HashMap<String, String> prefixTables` maps the unique prefix for each namespace to its associated namespace URI. The prefix structure will have only one entry since each namespace has one and only one prefix. A possible simplification of this would be to add the prefix as an attribute of the *URI*Table, but being consistent with the data-structures helps keep thing clear.

The Table value within the *URITables* `HashMap` contains the local-name values and global-values string tables for the namespace (*StringTable*). Because each namespace will contain many local-names, another `HashMap` of local values is contained in the *URITables* map (*localValueTables*).

```
private StringTable localNamesStringTable
private HashMap<String, StringTable> localValueTables
private StringTable globalStringTable
```

Using the introduced *StringTable* structure, each namespace Table structure has access to all local values by both string value content and index value. Note that the *globalStringTable* is its own string table outside of the *localValuesTable* since each namespace has only one global table shared by the entire namespace. Each new local-name encountered is mapped to the *localValuesTables* as `<"localName", new StringTable()>`.

c. Predefined EXI String Table Initialization Entries

The default namespaces for all EXI encoding are no namespace (""), the XML namespace ("xml"), and schema namespace ("xsi"). Each of their string table partitions are pre populated with local-names that are likely to occur within their namespace. Another namespace is only added if the XML document has an associated schema, in which case the schema instance namespace ("xsd") is also added by default. All other namespaces are either learned while processing the XML document or from the supporting schema.

The URI table entries are set to the values in Table 33 and pictorially in Figure 34. Note the ID 3 entry is only present if a schema is used in support of the XML document.

Partition	String ID	String Value
URI	0	"" [empty string]
URI	1	"http://www.w3.org/XML/1998/namespace"
URI	2	"http://www.w3.org/2001/XMLSchema-instance"
URI	3	"http://www.w3.org/2001/XMLSchema"

Table 33. EXI URI String Table Partition Initial Entries (After W3C, 2008)



Figure 34. EXI URI String Table Partition Initial Entries (From W3C, 2008)

Table 34 and Figure 35 list the initial entries for the prefix table partitions. Note, that each entry in the prefix table has ID 0 because it is the first and only prefix associated to that namespace URI. Each URI has one and only one unique prefix.

Partition	String ID	String Value
“”	0	""
XML-NS	0	“xml”
XSI-NS	0	“xsi”
XSD-NS	0	“xsd”

Table 34. EXI Prefix String Table Partition Initial Entries (After W3C, 2008)

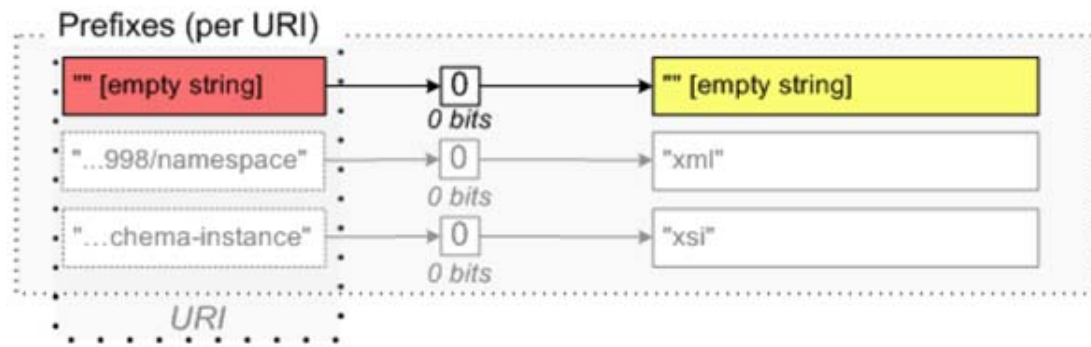


Figure 35. EXI Prefix String Table Partition Initial Entries (From W3C, 2008)

Table 35 through Table 37 define the default initial local-name entries for each of the default namespaces. The XML-NS contains 4 initial entries, the XSI-NS contains 2, and the XSD-NS contains 46. These entries are the most common local-names found for their corresponding namespace domains. For instance, a comparison of the XML Schema and the XSD namespace table entries reveals that the table's contents are all the reserved words of schema. This ensures a compact identifier is always present for known keywords without requiring a string literal encoding.

Partition	String ID	String Value
XML-NS	0	"space"
XML-NS	1	"lang"
XML-NS	2	"id"
XML-NS	3	"base"

Table 35. EXI Default XML Namespace String Table Local-Name Entries (From W3C, 2008)

Partition	String ID	String Value
XSI-NS	0	"type"
XSI-NS	1	"nil"

Table 36. Default XSI Namespace String Table Local-Name Entries (From W3C, 2008)

Partition	String ID	String Value
XSD-NS	0	"anyType"
XSD-NS	1	"anySimpleType"
XSD-NS	2	"string"
XSD-NS	3	"normalizedString"
XSD-NS	4	"token"
XSD-NS	5	"language"
XSD-NS	6	"Name"
XSD-NS	7	"NCName"
XSD-NS	8	"ID"
XSD-NS	9	"IDREF"
XSD-NS	10	"IDREFS"
XSD-NS	11	"ENTITY"
XSD-NS	12	"ENTITIES"
XSD-NS	13	"NMTOKEN"
XSD-NS	14	"NMTOKENS"
XSD-NS	15	"duration"
XSD-NS	16	"dateTime"
XSD-NS	17	"time"
XSD-NS	18	"date"
XSD-NS	19	"gYearMonth"
XSD-NS	20	"gYear"
XSD-NS	21	"gMonthDay"
XSD-NS	22	"gDay"
XSD-NS	23	"gMonth"
XSD-NS	24	"boolean"
XSD-NS	25	"base64Binary"
XSD-NS	26	"hexBinary"
XSD-NS	27	"float"
XSD-NS	28	"double"
XSD-NS	29	"anyURI"
XSD-NS	30	"QName"
XSD-NS	31	"NOTATION"
XSD-NS	32	"decimal"
XSD-NS	33	"integer"
XSD-NS	34	"nonPositiveInteger"
XSD-NS	35	"negativeInteger"
XSD-NS	36	"long"
XSD-NS	37	"int"
XSD-NS	38	"short"
XSD-NS	39	"byte"
XSD-NS	40	"nonNegativeInteger"
XSD-NS	41	"positiveInteger"
XSD-NS	42	"unsignedLong"
XSD-NS	43	"unsignedInt"
XSD-NS	44	"unsignedShort"
XSD-NS	45	"unsignedByte"

Table 37. Default XSD Namespace String Table Local-name Entries
(From W3C, 2008)

d. Default Event Mapping to String Table Entries

The EXI event mapping to string table partitions entry is listed in Table 38. Each XML event's contents are added to string table partition listed using the default datatype in the absence of a supporting document schema or datatype mappings. Important to note from this table is not all events content's are added to the string tables.

Content Item	Used In	Default Datatype	String Table Partition
indicator	NS	Boolean	
name	PI, DT, ER	String	
prefix	NS	String	prefix
public	DT	String	
qname	SE, AT	Qname	URI, LocalName
system	DT	String	
text	CM, PI	String	
uri	NS	String	URI
value	CH, AT	String	Value

Table 38. EXI Default XML Event Mapping to String Table Partition (From W3C, 2008)

A few special notes about Table 38:

- Value content items can be defined by a schema or datatype map to be other than the type string.
- All pruneable events, other than Namespace events are not added to the string tables and written to the EXI stream in raw ASCII format: Comments (CM), Processing Instructions (PI), DOCTYPE (DT), Entity (ER)
- Those events that are not added are always written to the EXI stream as ASCII text without compact identifier.
- Qname (Qualified Name) is a special String type.

e. Local-name String Found

When an event's string value is found in its namespace string table partitions local-names string table, the string is represented as zero (0) encoded as an Unsigned Integer followed by its string table entries compact identifier, that is, table index. The compact identifier of the string value is encoded as an n-bit unsigned integer, where n is the ceiling value $\lceil \log_2 m \rceil$, and m is the number of entries in the string table partition at the time of the current operation. The concept of "at the time of the current operation" is important as it indicates the ability to grow or learn as the XML document is processed. This implies a growing n-bit length as the number of entries within each string table partition increases. An entry that was initially encoded with only 3 bits may later be encoded with the same numerical value but by using 5 bits if more strings having been added to the string table.

f. String Value Found in Global Only

When a string value is found in the global value partition, but not in its local-name value partition, the string value is represented as one encoded as an Unsigned Integer followed by the compact identifier of the string value from within the namespace's global table partition. The compact identifier is encoded as an n-bit unsigned integer, where n is $\lceil \log_2 m \rceil$ and m is the number of entries in the associated global partition at the time of the operation, identical to that of the local-name string table hit just discussed. That string is then added to its local-name values table and assigned the next sequential compact identifier.

This case of a local-name miss and global hit can arise if the namespace had the same value previously from a different local-name, but the current local-name has yet to come across the same string value. Remember, the namespace global table is shared by all local-names of the namespace so any occurrence within the namespace, and is visible to all other local-name within the namespace.

g. String Not Found (Names and Values)

When a string is not found anywhere within the namespace string tables, its encoding depends on whether the string is a local-name (element or attribute name) or value (attribute value or element content).

If the string is a local-name, the string is encoded as a string literal after an unsigned integer of value $L + 1$, where L is the character length of the string. After encoding the string to the EXI stream, the string is added to its namespace's local-name string table, which will require the creation of a new local-name table, and is assigned the next sequential compact identifier.

If the string is an element content or attribute value and not found within either the global string table or the value string table for its associated local-name within the namespace, it is encoded as a string literal after an unsigned integer of value $L + 2$, where L is the string character length of the string.

After encoding the string value to the EXI stream, if and only if L is less than the `valueMaxLength` option value as defined in the header, and the number of values within the table is less than `maxValueCapacity` as defined in the header, the string is added to both its associated local-name's value string table and the global string table for that namespace.

h. URI String Table Found

If a URI's value is found in the namespace string table, its compact identifier is the index to the URI within the URI string table plus 1. If the URI is not found, its compact identifier is 0. Both compact identifiers are coded with enough bits based on the total number of entries currently in the URI table list + 1. Note that a URI miss is index 0, and that there is no namespace URI at index 0 due to the + 1 offset. A URI hit is always +1 so there will be no conflict in indexing and the URI table is allowed to grow.

i. Value Scope

A note about values that is not immediately obvious is that the value entries are scoped to their local-name scope within the XML document. For example from the notebook.xml example, Table 40 (three pages down for quick reference) the attribute date has document-wide scope because it is first declared in the root element, and so it retains all values of date attributes anywhere within the XML document. The element <subject> however, drops out of scope from its first occurrence to the second.

Element subject, because of value content scoping, the first character value “EXI” is not retained in the local hit list. At the end of EXI processing, the <subject> element values list consist of only a single entry “shopping list” without reference to the earlier “EXI”. This scope process of the EXI string tables is made more apparent in the examples shown in Figure 36 and Figure 37. The first shows that each namespace has its own unique set of local-names; a total of six localNames are all associated to the no namespace or empty namespace table. The second expands on the relation between local values and the global values entries, but at the completion of EXI generation. Note that the local-name values do not index to the global as the arrows are presented only to point out the common entries in both.

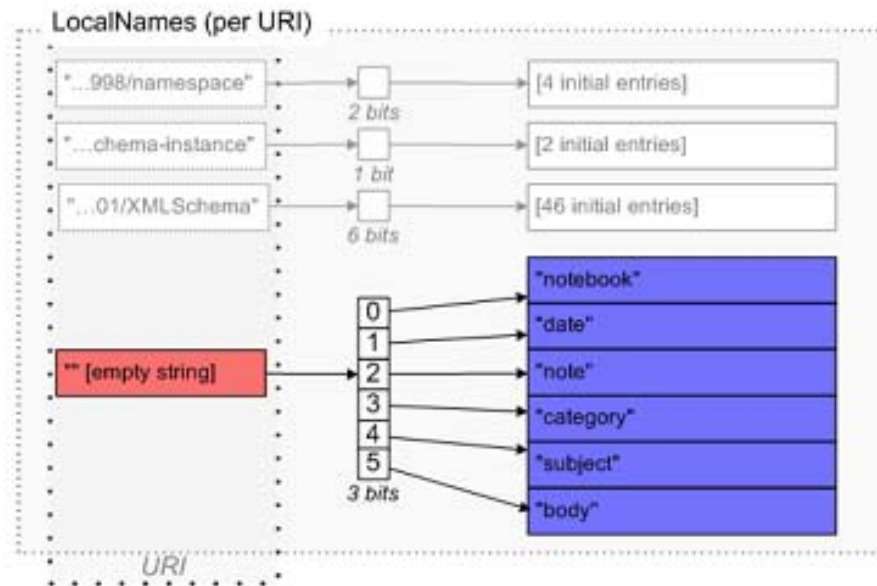


Figure 36. Local-Name String Table Entries Based on Notebook.xml Example
(From W3C, 2008)

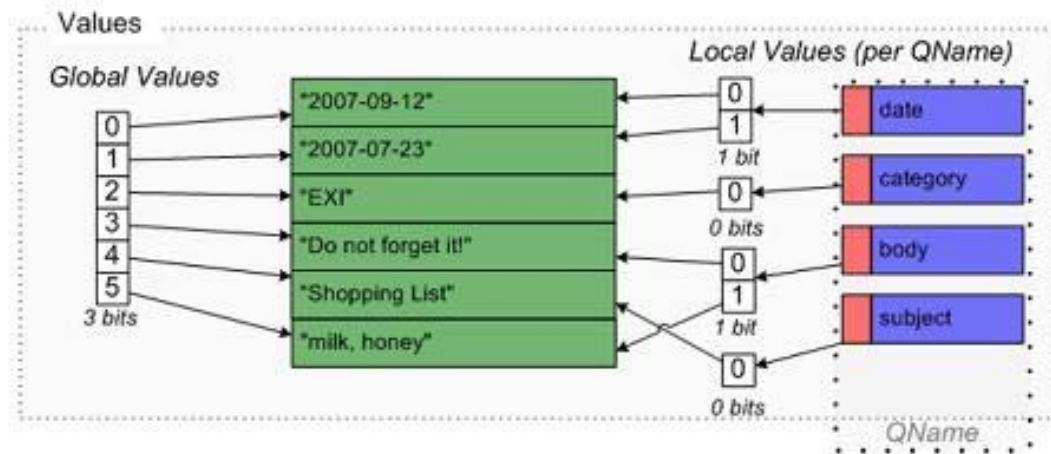


Figure 37. Global and Local Values String Table Entry Mapping to Local-Name
(From W3C, 2008)

Another example of the value scope rules is contained in the customer.xml sample below. In this example, the value cust201, in bold, is repeated three times, but at best each occurrence results in a global hit.

1) The first occurrence is within the <customer> element for attribute custID. This first occurrence results in both a local and global miss, and so puts the value in the global table as well as the custID local-name table.

2) The second occurrence is within the first <order> element as the value of attribute orderBy. This occurrence results in a global hit from the <customer> element custID attribute entry, but not a local hit. The orderBy local-name values table is then updated with the value "cust201."

3) The first <order> element fires its EndElement event before any additional occurrences of the value cust201. The <order> element and subcomponents of it are now out of scope and so lose reference to their values; the string values are deleted from the string table.

4) The second occurrence of the <order> element's orderBy attribute value cust201 does not result in a local hit even though it was previously declared due to value scope limits. The first order element has fired its end-element event, terminating value scope. It does however result in a global hit, but this hit is based on the original occurrence from within the <customer> element's custID attribute.

```
<?xml version="1.0" encoding="UTF-8"?>
<customers>
  <customer custID="cust201" custType="home">
    <orders>
      <order orderID="or10311" orderBy="cust201">
        <orderDate>8/1/2008</orderDate>
      </order>
      <order orderID="or10311" orderBy="cust201">
        <orderDate>8/1/2008</orderDate>
      </order>
    </orders>
  </customer>
</customers>
```

j. OPENER-EXI String Table Example

A more interesting view of this process is found in Table 39, output listing from the OPENER-EXI implementation. It demonstrates the building of the string table for the EXI “Hello World” notebook.xml. Referencing to the immediately following Table 40, the notebook.xml for quick reference, it can be seen how the mapping of values, local-names and namespaces to string tables is accomplished using the HashMap design presented in this chapter section. Because the notebook.xml example only uses the no namespace, the XML and XSI string tables are empty.

<pre>----- NAMESPACE TABLES -> [PREFIX =], [URI =] ----- QNames (for this URI) 0 notebook 1 date 2 note 3 category 4 subject 5 body VALUES (for this LOCALNAME of this URI) notebook date 0 2007-09-12 1 2007-07-23 note category 0 EXI subject 0 shopping list body 0 Do not forget it! 1 milk, honey GLOBALS 0 2007-09-12 1 2007-07-23 2 EXI 3 Do not forget it! 4 shopping list 5 milk, honey</pre>
<pre>----- NAMESPACE TABLES -> [PREFIX = XML], [URI = HTTP://WWW.W3.ORG/XML/1998/NAMESPACE] ----- QNames (for this URI) VALUES (for this LOCALNAME of this URI) GLOBALS</pre>
<pre>----- NAMESPACE TABLES -> [PREFIX = XSI], [URI = HTTP://WWW.W3.ORG/2001/XMLSCHEMA-INSTANCE] ----- QNames (for this URI) VALUES (for this LOCALNAME of this URI) GLOBALS</pre>

Table 39. OPENER-EXI (Notebook.xml) String Table Build Example Output

```

<?xml version="1.0" encoding="UTF-8"?>
<notebook date="2007-09-12">
  <note date="2007-07-23" category="EXI">
    <subject>EXI</subject>
    <body>Do not forget it!</body>
  </note>
  <note date="2007-09-12">
    <subject>shopping list</subject>
    <body>milk, honey</body>
  </note>
</notebook>

```

Table 40. The notebook.xml Local Copy (From W3C, 2008)

From the previous tables it is clear to see that each element and attribute of the notebook example is listed within the correct namespace, within its respective local-name string table portions and each of their values is contained in the values section. The global section contains all values found within the document for the empty namespace in order of document occurrence as required. The general algorithm for the creation of the string table entries is contained in Table 41.

```

For each event within the XML document (likely a SAX event)
  Get the event namespace
  Get the event prefix
  Get the event Qname
  Get the event value

  Index into string table with namespace
  Index into the namespace table with prefix
  Index into the prefix table with Qname
  Index into the Qname table with value

  If value exist then
    return 0 + associated index
  else if value exist in Global
    return 1 + associated global index
  else
    return stringLength + [1,2]+ String
    if stringLength <= valueMaxLength && valueAmount <
      valuePartitionCapacity
      add string to global
      add string to values table ([namespace][prefix][Qname])
      valueAmount++

```

Table 41. String Table Creation Pseudocode Algorithm

2. Grammars and Events

Grammars and Events are the EXI's mechanism to map native XML into a compact and efficient binary format. In general, events define the underlying XML document's content and characteristics, and grammars define the underlying XML structure by means of a list of the events contained within the scope of the grammar. Grammars in the rawest form equate to a start-element and retain scope until the matching end-element within the input XML document.

a. Information Grammar Theory (Chomsky)

Both grammars and events are learned for each XML document by means of a supporting schema or by processing the XML document. The learning process is similar to Chomsky grammars, a hierarchical-based formal grammar for defining a language (Chomsky, 1956). To summarize the Chomsky concept as it pertains to EXI:

- It consists of a finite (countable) set of starting symbols. In EXI terms these are XML events, which are known beforehand from SAX or another XML parser. XML has a countable number of event types.
- Production rules are applied to the finite starting symbols to define an infinite language. In EXI terms, each XML event can take on an infinite set of values and each XML event has its own rules for processing. For example, an element event can have an infinite set of values, even for the same element name.
- Sequences of symbols can derive new symbols. In EXI terms, grammars contain one or more events, growing as the document is processed and based on the structure of the document. EXI grammars are the new symbols derived for sequences of symbols.
- A starting symbol derivation ends at terminal symbols. In EXI terms, grammars are terminated by an end-element event that match the same start-element.

Grammar theory is a sophisticated, powerful and essential part of all computer languages (Davis & Weyuker, 1994). This theory is the basis of computing today and is what enables all computer interactions. EXI is no exception to this, but unlike most applications, EXI is coded directly with formal grammars instead of using a higher-level language API. This is a key point that is not immediately noticed, but important to realize: EXI itself is not an API, but an algorithm definition, and therefore the grammar process must be directly implemented.

b. Events

EXI events are really nothing more than XML events, or stated another way EXI events that define the XML structural characteristics and content of the input XML document. These events are the basis of the EXI formal grammar process, compactness and efficiencies. These capabilities are accomplished by encoding event's information utilizing a compact identification code based on the current grammar, and represents their XML content when supported with a schema definition with a binary representation. An overview of the available EXI events and XML information they describe is listed in Table 42.

EXI Event Type		Event Body		Event Notation
		Characteristic	Content	
Start Document				SD
End Document				ED
Start-element		qname		SE(qname)
				SE(*)
				SE(uri: *)
End-element				EE
Attribute		qname	value	AT(qname)
				AT(*)
Characters			value	CH
Namespace Declaration	(+)	uri, prefix		NS
Processing Instruction	(+)	name, text		PI
DOCTYPE	(+)	name, public, system, text		DT
Entity Reference	(+)	name		ER
Comment	(+)	Text		CM
(+) indicated event can be pruned (ignorable) with options				

Table 42. EXI Defined Event Types and Notation (From W3C, 2008)

c. Event Codes

Event codes are a sequence of 1 to 3 non-negative integers called parts that uniquely identify one event from another. The number of parts depends on several aspects: the likelihood of the particular event type with regards to the general XML information set, which EXI pruning options are enabled, and the current grammar. Table 43 lists the possible events allowed by grammar type, and how the events' codes are assigned.

DocContent <i>(Document scope)</i>	Document: SD DocContent 0 DocContent: SE (*) DocEnd 0 DT DocContent 1.0 CM DocContent 1.1.0 PI DocContent 1.1.1 DocEnd: ED 0 CM DocEnd 1.0 PI DocEnd 1.1
FragmentContent <i>(Fragment scope)</i>	Fragment: SD FragmentContent 0 FragmentContent: SE (*) FragmentContent 0 ED 1 CM FragmentContent 2.0 PI FragmentContent 2.1
StartTagContent <i>(Element start scope)</i>	StartTagContent: EE 0.0 AT (*) StartTagContent 0.1 NS StartTagContent 0.2 SC Fragment 0.3 (SE) ChildContentItems (0.4) (CH)
ElementContent <i>(Element scope)</i>	ElementContent: EE 0 ChildContentItems (1.0) ChildContentItems (n.m): SE (*) ElementContent n.m CH ElementContent n.(m+1) ER ElementContent n.(m+2) CM ElementContent n.(m+3).0 PI ElementContent n.(m+3).1

Table 43. EXI Grammars by Events Structure (After W3C, 2008)

The explanation of the coding of Table 43:

- Event code part 1 is the count of events within the grammar plus the part 1 constant listed in Table 43. All events will have at least part 1 set, with other parts optional depending on EXI pruning options.
- Event code part 2 uniquely identifies the event type. On the first occurrence of an event within a grammar, all events, other than document wide one time only events, have part 2 set. The part 2 portion of the event code can be minimized through event pruning options.
- Event code part 3 is used only for highly unlikely and pruneable events. Part 3 is seldom used in the long run average of XML events.
- Pruned Events are those events that can be ignored by the EXI processor. If the EXI processor is pruning events, those pruned events are removed from the event code sequence within Table 43, and all lower events move up one position. For example, if the current grammar is ElementContent, with ER and PI events pruned, but CM preserved, then the new baseline grammar event code assignments would be:
SE (*) ElementContent n.m
CH ElementContent n.(m+1)
CM ElementContent n.(m+2)

Note that the part 3 code portion is not used, and that the CM event now has its part 2 set to m+1 instead of (m+3).0 because the two preceding pruneable events are ignored moving CM up two positions to m+1.

d. Bit and Byte Representation of Events Codes

In general, the fewest number of bits needed to encode any of the parts of an event code is found by taking the ceiling $\lceil \log_2 d \rceil = n$, where d is the number of events in the current Grammar. If byte alignment or pre compression is used, each part of the event is encoded with the fewest number of bytes $\left\lceil \frac{\log_2 d}{8} \right\rceil = n$.

Regardless of options settings, if there is only one distinct value for the first part of an event, equating to 0.m.c, where the first part is omitted because the event is known by implication. That is, given that there is only one possibility for the event based on the grammar rules, the event is known without its code because it is the only possible event. This is confirmed by the fact $\lceil \log_2 1 \rceil = 0$.

The elementary example presented in Table 44 demonstrates a document that has 4 elements. In this example using bit-alignment, the first part of each event is in the range of 0 to 4, or a total of 5 values. This requires $\lceil \log_2 5 \rceil = 3$ bits to encode the first part of the event code, the second part $\lceil \log_2 4 \rceil = 2$ and the third part, $\lceil \log_2 2 \rceil = 1$. A total of only 6 bits are needed to encode all the required event codes. However, if a byte alignment is used, a higher total of 24 bits or 3 bytes are needed, one byte for each part. Table 45 gives a verbose comparison of the bit and byte aligned event code sizes.

DocContent	Event code
SE("A") DocEnd	0
SE("B") DocEnd	1
SE("C") DocEnd	2
SE("D") DocEnd	3
SE(*) DocEnd	4.0
DT DocContent	4.1
CH DocContent	4.2
CM DocContent	4.3.0
PI DocContent	4.3.1

Table 44. Elementary Event Codes Example (From W3C, 2008)

Event	Part			Bit Encoding	# Bits	Byte Encoding	# Bits	# Bytes
SE("A")	0			000	3	00000000	8	1
SE("B")	1			001	3	00000001	8	1
SE("C")	2			010	3	00000010	8	1
SE("D")	3			011	3	00000011	8	1
SE(*)	4	0		100 00	5	00000100 00000000	16	2
DT	4	1		100 01	5	00000100 00000001	16	2
CH	4	2		100 10	5	00000100 00000010	16	2
CM	4	3	0	100 11 0	6	00000100 00000010 00000000	24	3
PI	4	3	1	100 11 1	6	00000100 00000010 00000001	24	3
	5	4	2	# distinct values (d)				
	3	2	1	# bits per part $\lceil \log_2 d \rceil$				
	1	1	1	# bytes per part $\lceil (\log_2 d) / 8 \rceil$				

Table 45. Verbose Comparison of Event Codes between Bit and Byte Alignments
(From W3C, 2008)

e. Repeating Event and Schema Impact on Event Codes

The event codes demonstrated in Table 44 and Table 45 are presented for encodings of the first time the event is encountered, and in schemaless mode. All follow-on occurrences of these same events, regardless of schema-informed or schemaless, are be encoded with a single integer, event code part 1 only, equal to the index of that event's location within the current grammar's event list.

This is where schemaless and schema-informed technique diverges for event-code assignment. A schema-informed encoding knows beforehand exactly which events will occur within any particular grammar before processing the XML documents, and so, only use a single integer, event code part 1, to represent all SE, AT, CH, EE and NS events. However, even with a schema-informed encoding, the unlikely events such as PI and CM can still be encoded with up to 3 integer event code parts depending on pruning options.

f. Events Implementation Notes and Lessons Learned

EXI events are most likely derived by using a XML parser, such as Java's Simple API for XML (SAX) to process the XML document. SAX is part of the Java Standard Edition (SE) library, and is used in the OPENER-EXI code set implementation. The SAX parser sequentially reads an XML document firing XML events for each XML structure that is contained within the documents in the order they appear within the XML document; left to right and top to bottom.

There are a few unique aspects of the SAX parser that need to be managed to ensure proper EXI processing:

1. SAX Namespace events, though enclosed within an XML element and after the element declaration, are fired before the encapsulating element's event. For example, the foo namespace event fires before the personnel start-element event based on the XML code sample `<ns:personnel xmlns:ns="urn:foo">`.

This order of event firing is opposite of EXI namespace processing. EXI namespaces are declared as member of the element and are processed after the encapsulating element's event. To accommodate this disparity, special-case handling needs to be implemented. OPENER-EXI places all namespace events in a document ordered container until the next start-element event is fired, knowing the next start-element event is the parent element of all previous namespaces. After the start-element event fires and is processed, the container of namespaces is processed.

2. SAX Character events are fired without indication of parent element association. That is, the character content of an element is fired without reference to the element, but EXI needs the parent reference. OPENER-EXI's solution to this problem is to place all start-element events into a stack as they occur, and then pop them from the stack with each end-element event. When a character event occurs, a peek at the stack's head indicates the parent element associated for the character event. Knowing the well-formed XML document rules, this method maintains synchronization with character content and parent element relations.

3. Multi-line character (CH) and comment (CM) events fire a new event for each line of the character or comment field. The comment example below, assuming within an ElementContent grammar and preserving all pruneable events, fires two distinct comment events, both with the event code of $n.(m+3).0$. Referring back to Table 42, comment events are not added to the string table or grammar. They will have the same event code because the grammar size does not change for comment events, assuming back-to-back events as in this example; the n of the event code $\lceil \log_2 n \rceil$ does not change between events.

```
<!--
Comment line 1
Comment line 2
-->
```

4. The output EXI stream is built off of XML events in the same order as they are present within the input document. For example the first element within the notebook.xml example (`<notebook date="2007-09-12">`) would generate events in order:

1. SE for the “notebook” tag
2. AT for the “date” attribute of the notebook element, which also contains the date value content “2007-09-12”

g. Grammar Creation

The purpose of the grammar-creation process is to create the structure of the XML documents in processor memory for mapping XML into EXI format and EXI to XML. The idea is the first time an event occurs, it is recorded into a grammar, and each subsequent occurrence of that same event can then be compactly and efficiently mapped based on the EXI grammar structure in memory. This is the “learned” structural processing that EXI employs that has enabled its ability to realize high levels of compactness. Constructed grammars correspond to the XML structure and the rule set for EXI event codes as listed in Table 43.

Each event fired from the XML parser is checked to the current grammar's list of events to find its assigned event-code representation. This code, event number within the grammar's event list is written to file, which is shorter than the event native XML tagging, followed by that event's item contents. If there is no matching event in the grammar's event list, the event is added to the grammar's event list, event code assigned accordingly and written to the EXI stream.

New grammars are created for each unique start-element event within the XML document. The EXI processor maintains lists of grammars using a stack data-structure as well as a HashMap. The grammar lists are pre-populated based on the supporting schema or learned while processing the XML document or both.

Each start-element event causes several EXI transitions:

1. The event is processed under the current grammar
2. The current grammar is pushed onto the grammar stack
3. The event is written to the EXI stream
4. The grammar HashMap is searched for a matching grammar for the start-element event
5. The current grammar is set to either the returned grammar from the HashMap search or the new grammar created for the start-element

Grammars have scope, based on which start-element created the grammar from within the XML document and its termination when the matching end-element XML event is fired. A stack of grammars is a simple mechanism to help maintain grammar scope and is the technique used in OPENER-EXI.

h. Grammar Document Processing

An initial schemaless EXI processing grammar state is a single DocContent grammar as the current and only grammar. The DocContent level refers to all events that have document wide scope such as the root element of an XML document and processing instructions. As the XML document is processed, the root element will

trigger the creation of a new grammar. The subsequent grammar from the DocContent for the root element will be a StartTagContent grammar followed by its ElementContent grammar.

There are a few points to highlight about grammars:

1. Valid XML documents can have only a single root element. A single root for the document implies that the DocContent grammar will have one and only one SE event.

2. Pruneable events (other than NS) are not added to the grammar list. This is indicated in Table 43 and discussed in the multiline comment events example previously presented. Pruneable events will always have the same part 2 and part 3 event codes, but part 1 may change depending on the size, content count, of the grammar. Part 1 will always be the size of the current grammar list plus 1 for pruneable events.

3. All start-element events equate to a StartTagContent and ElementContent grammars, with all valid XML documents having a matching end-element. Based on this valid XML document rule, all ElementContent grammar by default contains its EE event at grammar creation.

A pictorial example of this Grammar learning process from the EXI specification is Figure 38. Within this figure, the layers, namely start-element events, represent the scope of each grammar based on the notebook.xml example. The lowest layer is the DocContent, followed by the element grammar notebook, the root element, which is followed by the element grammar note. The pertinent notebook.xml code for this example:

```
<notebook date="2007-09-12">
  <note date="2007-07-23" category="EXI">
    <subject>EXI</subject>
    <body>Do not forget it!</body>
```

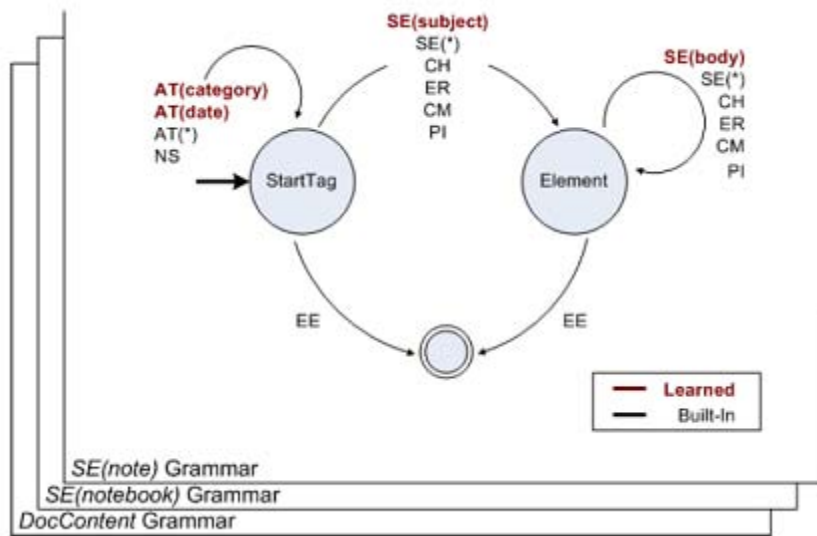



Figure 38. EXI Grammar Learning, Discovery and Transition Processes Based on the notebook.xml Document (From W3C, 2008)

Reviewing the pertinent code and Figure 38, the building of the note element grammars, top layer, takes shape. “Note” StatTagContent grammar has attributes “date” and “category” and also has sub elements named “subject” and “body.” “Subject” is defined as occurring within the “note” scope, although the grammar for “subject” is not created, no layer SE(subject) Grammar until processing the start tag of “subject.” The “subject” element qname is a part of the “note” grammar, but the “subject” element’s contents (attributes, comments, and character data) are part of the “subject” StartTagContent grammar after it is created. It is important to realize that the attributes are processed before any subsequent elements for an element, that is, events are sequentially processed from left to right and top to bottom. The listed sequence is essential for the EXI process.

A more general view of the grammar process is contained in Figure 39. Here, as events are processed by the XML Parser, Java SAX for example, they are passed to the current grammar, such as DocContent at run start. The new event is first added to the existing grammar if the event is an addable type. Then, if the new event is a start-element (SE) event a new or existing StartTagContent grammar is created or returned.

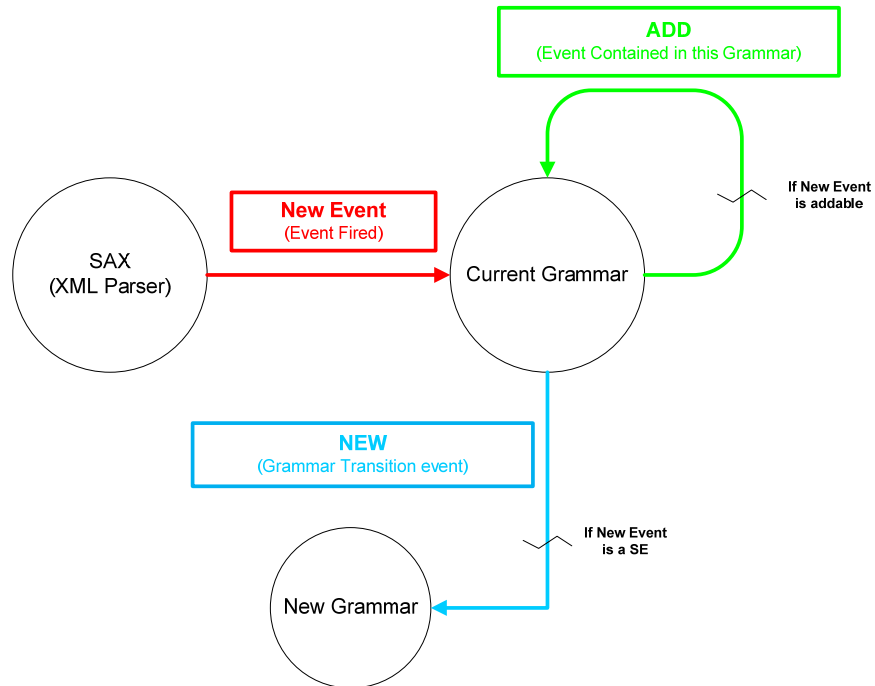


Figure 39. Abstract EXI Grammar Creation Process Diagram

Based on Figure 38, the “note” element contains two other elements, “subject” and “body.” Subject is a SE(*) event of “note” generating Note_StartTagContent grammar and body is a SE(*) of the Note_ElementContent grammar, but only after the complete processing of the subject_StartTagContent grammar. Within each of the “subject” and “body” StartTagContent grammars, subsequent events are added to each, such as their character events (CH).

The transition from one grammar to another is depicted in Discrete Event Graph (DEG) notation within Figure 40. The framework of a DEG provides the relationship between states (EXI Grammars) that a system, the EXI processor, can be in and the activity of the system (Buss, 2009). All DEG consist of nodes and directed edges, where a node (circle) within EXI is a Grammar, and an edge (line) is an EXI event. One of the motivations for this type of representation is that the system can be in 1 and only 1 state or node at a given time, but can have multiple edges or events scheduled from each node.

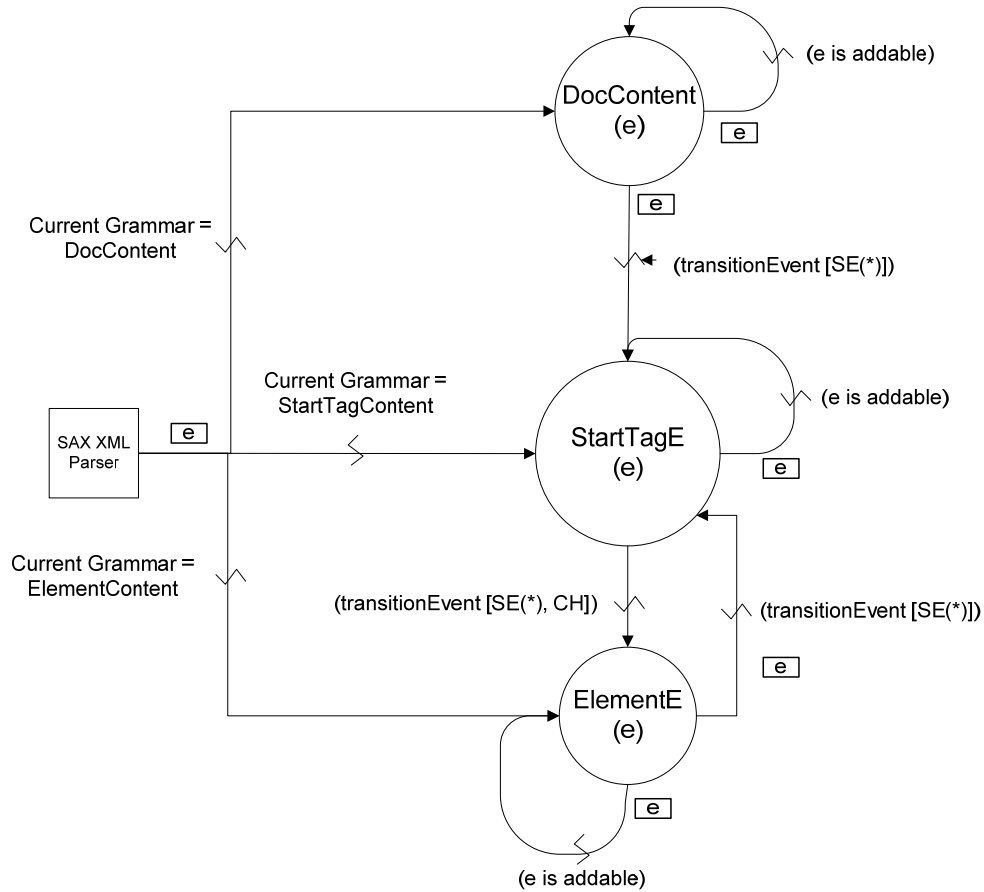


Figure 40. Abstract EXI Grammar Transition Process

EXI is always within a grammar state and transitions to another state triggered by EXI events, such as new Start-element events. Depending on the current grammar state and the current event, the EXI processor will remain within its current grammar or transition to a new grammar.

Reviewing the StartTagE grammar node in Figure 40, a SE(*) event triggers 3 processes:

1. Adding a event to the current StartTagE grammar
2. The creation of an ElementE grammar based on the current grammar
3. The creation of a new StartTagE grammar for the new event and transitioning or setting this new grammar to be the current grammar.

Further review of the layered structure of the grammars, along with knowledge about XML's structure leads to imply a stack like structuring control of grammars. As the depth into the XML document increases, the previous-depth grammar is pushed onto the stack. Of course, as the scope of a grammar expires, the termination of the element by its end-element event cause the stack to pop to get the previous depth's grammar. A stack is used within OPENER-EXI to maintain grammar scope.

Overall, grammars create the XML structure by defining what information may appear at any particular depth of the XML document. This information is used to encode the XML document's content with the fewest bits needed in order to recover the content as well as structure at decoding. Grammars are the map, and the events are the contents. Figure 41 is a simple graphic that overviews this basic process based on the notebook.xml example. The content of the XML document is parsed into its events, gray circles, and content, colored circles, and then written to the EXI stream in document order.

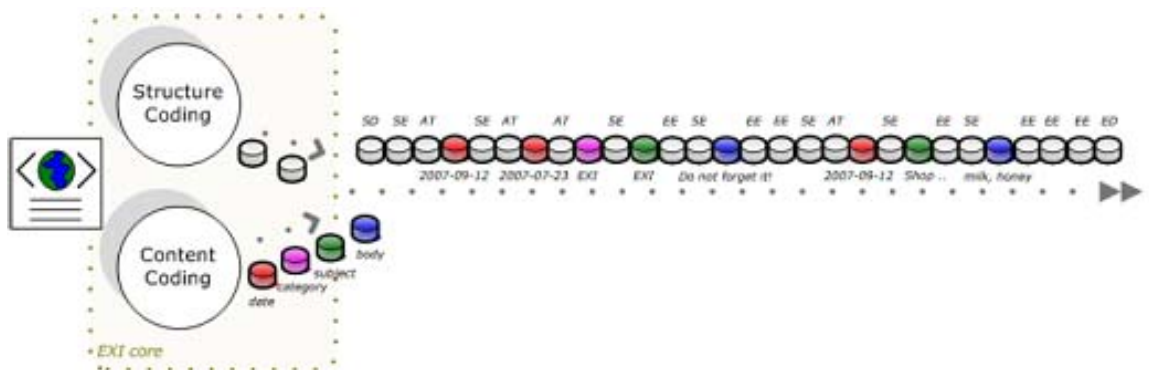


Figure 41. Color Coded Overview of notebook.xml Grammar/Event Encoding
(From W3C, 2008)

i. Schema Grammar Building

Much like pre populating the string tables based on the supporting schema definition, grammars are also initialized and the grammar HashMap filled with empty grammars based on the supporting schema. Parsing the schema extracts the elements that are defined for the document. For each defined element, a grammar is created for that element and added it to the grammar HashMap. Regardless whether there is a schema or not, a DocContent is the initial grammar and is loaded on the stack immediately at EXI initialization.

A note about the usage of schemas is that the XML document must be schema valid, i.e., lexigraphically the same as the schema that defines it. The reason is XML document ordering of elements and attributes is loosely structured, but when processing for EXI compression, the order is essential as the compact identifier process is a sequentially numbered process based on event occurrence within the document. The Table 46 example demonstrates this concept with two valid XML fragments based on the provided schema, but each XML fragment would generate distinctly different EXI outputs due to attribute ordering.

Scheam “entry” Definition	<code><xs:element name="entry"> <xs:attribute name="date" type="xs:date"/> <xs:attribute name="time" type="xs:string"/> </xs:element></code>
XML Fragment 1	<code><entry date="2007-09-12" time="12:00"></code>
XML Fragment 2	<code><entry time="12:00" date=" 2007-09-12"></code>

Table 46. Rigid Schema Formatting Compliance Requirment for XML Documments Example Case

Both of these XML fragments, rows 2 and 3 of Table 46, are the same in terms of XML information based on the schema, row 1, and both would generate valid XML documents as well as be valid against the schema. The order of attributes is not important in terms of XML schema, only that the defined attributes exist and that no undefined attributes exist. Problems arise when performing EXI encoding or decoding. Given that EXI grammars are built sequentially in schema order, the occurrence of the

second fragment results in an EXI error because the grammar for fragments is defined as date (datatype date) followed by time (datatype string), but the second fragment has the typed attributes in reverse order.

In some cases, this may not result in fatal errors such as when the datatype is string only. However, the W3C (2007) *Best Practice* document states “All bets are off” if the XML document is not in full compliance with the schema structure. In general, a schema can deliver higher levels of compactness, but the document must be in lexical order with the schema ordering to ensure valid round-trip coding. Such consideration of ordering is typically performed as part of the EXI encoding process.

j. Verbose Event and Grammar Encoding of Notebook.xml

An OPENER-EXI event-by-event constructing of the notebook.xml document is defined below. The start of each new line, left justified NOT indented represents the fired event from the XML document with the name of the event type in parentheses and its assigned event code in brackets. The grammar built with each event is on the immediately following indented line. The grammar type, docContent, StartTag and Element, are pre-fixed with the grammar name, which is the start-element name. The grammar’s list of contents is within the square brackets immediately after the grammar name on the same line as the grammar name. The vertical line in the middle of the square brackets separates the sections of the grammar contents: docContent | endDoc, and StartTagContent | ElementContent.

All preservation options were set to true for this encoding, though only the NS event affected the encoding in terms of event codes because no comments, processing instructions, entities or document types are contained within the notebook.xml example.

```
startDocName (Start Document) [0]
  Grammar_DocContent:DocContent [ | ]
notebook (Start Element (any)) [0]
  Grammar_DocContent:DocContent [ | ]
date (Attribute (Any)) [0.1]
  Grammar_StartTagContent:notebook [ | ]
note (Start Element (any)) [1.3]
  Grammar_StartTagContent:notebook [date | ]
```

```

date (Attribute (Any)) [0.1]
    Grammar_StartTagContent:note [ | ]
category (Attribute (Any)) [1.1]
    Grammar_StartTagContent:note [date | ]
subject (Start Element (any)) [2.3]
    Grammar_StartTagContent:note [category, date | ]
characters (Characters) [0.4]
    Grammar_StartTagContent:subject [ | ]
subject (End Element) [0]
    Grammar_ElementContent:subject [characters | ]
body (Start Element (any)) [1.0]
    Grammar_ElementContent:note [subject, category, date | ]
characters (Characters) [0.4]
    Grammar_StartTagContent:body [ | ]
body (End Element) [0]
    Grammar_ElementContent:body [characters | ]
note (End Element) [1]
    Grammar_ElementContent:note [subject, category, date | body]
note (Start Element (any)) [1.0]
    Grammar_ElementContent:notebook [note, date | ]
date (Attribute (Any)) [2]
    Grammar_StartTagContent:note [subject, category, date | body]
subject (Start Element (any)) [0]
    Grammar_StartTagContent:note [subject, category, date | body]
characters (Characters) [0]
    Grammar_StartTagContent:subject [characters | ]
subject (End Element) [0]
    Grammar_ElementContent:subject [characters | ]
body (Start Element (any)) [0]
    Grammar_ElementContent:note [subject, category, date | body]
characters (Characters) [0]
    Grammar_StartTagContent:body [characters | ]
body (End Element) [0]
    Grammar_ElementContent:body [characters | ]
note (End Element) [1]
    Grammar_ElementContent:note [subject, category, date | body]
notebook (End Element) [1]
    Grammar_ElementContent:notebook [note, date | note]
endDocName (End Document) [0]
    Grammar_DocEnd:DocContent [notebook | ]

```

The takeaway from this algorithm description is to highlight that grammars are not focused on the values, but rather the structure of the XML file. The string tables retain all values and structural names of the XML document, and the grammar defines the XML document (EXI Stream) structure.

k. Verbose Byte-Aligned Encoding of Notebook.xml Example

A byte-aligned byte-by-byte verbose encoding of the notebook.xml example document follows. Here, each byte of the encoded XML document is displayed in its EXI binary format followed by its meaning and logic creation rules. The notebook.xml example document is reproduced for quick comparison reference in Table 47 immediately after the encoding. Note that all of the EXI preservation options were kept true in this example encoding so that the event codes would be directly transparent to the event codes as listed in Table 43. A general pseudocode algorithm for EXI decoding is listed in Table 48.

Document

1 (1001 0000) Header: EXI, non-final version with no options

<notebook>

2 (0000 0001)

DocContent

uri "" hit = 1

SD = 0 for 0 bits

SE(*) = 0 for 0 bits

3 (0000 1001)

LocalName miss "notebook" length + 1

4-11 ASCII

notebook

date="2007-09-12"

12 (0000 0001)

StartTagNotebook

AT(*) 0.1 with n=0 for 0 bits

13 (0000 0001)

uri "" hit = 1

14 (0000 0101)

localName miss "date" length + 1

15-18 ASCII

date

19 (0000 1100)

Value miss "2007-09-12" length + 2

20-29 ASCII

2007-09-12

<note>

30 (0000 0001)
31 (0000 0011)
32 (0000 0001)
33 (0000 0101)
34-37 ASCII

date="2007-07-23"

38 (0000 0001)
39 (0000 0001)
40 (0000 0000)
41 (0000 0001)

42 (0000 1100)
43-52 ASCII

category="EXI"

53 (0000 0001)
54 (0000 0001)
55 (0000 0001)
56 (0000 1001)
57-64 ASCII

65 (0000 0101)
66-68 ASCII

<subject>

69 (0000 0010)
70 (0000 0011)
71 (0000 0001)
72 (0000 1000)
73-79 ASCII

EXI

80 (0000 0100)
81 (0000 0001)
82 (0000 0010)

</subject>

83 (0000 0000)

StartTagNotebook

SE(*) of (1.3) [content count +1].3 (date)
uri "" hit = 1
localName miss "note" length + 1
note

StartTagNote

AT(*) 0.1 0 not encoded...implied
uri "" hit = 1
LocalName hit = 0
date entry is 1

Value miss "2007-07-23" length + 2
2007-07-23

StartTagNote

AT(*) 1.1 (date)
uri "" hit = 1
LocalName miss "category" length + 1
category

Values miss "EXI" length + 2
EXI

StartTagNote

SE(*) 2.3 [content count + 1].3 (category, date)
uri "" hit = 1
LocalName miss "subject" length + 1
subject

StartTagSubject

CH 0.4
Global hit = 2
EXI entry 2 (2007-09-12, 2007-07-23, EXI)

ElementSubject

EE 0 subject

<body >

84 (0000 0001)
85 (0000 0000)
86 (0000 0001)
87 (0000 0101)
88-91 ASCII

Do not forget it!

92 (0000 0011)
93 (0001 0011)
94-110 ASCII

ElementNote

SE(*) 1.0 content count +1.0
uri "" hit = 1
LocalName miss "body" length + 1
body

StartTagBody

CH 0.4
Values miss "Do not forget it!" length + 2
Do not forget it!

</body>

111 (0000 0000)

ElementBody

EE 0 body

</note>

112 (0000 0001)

ElementNote

EE 1 (see below note EE) (body)

<note>

113 (0000 0001)
114 (0000 0000)
115 (0000 0001)
116 (0000 0000)
117 (0000 0010)

ElementNotebook

SE 1.0 content count + 1.0
uri "" hit = 1
LocalName hit = 0
hit position = 2

date="2007-09-12"

118 (0000 0010)
119 (0000 0000)
120 (0000 0000)

StartTagNote

AT(date) 2 (subject, category, date)
Value hit = 0
hit position = 0 (2007-09-12, 2007-07-23)

<subject>

121 (0000 0000)

StartTagNote

SE(subject) (subject, category, date)

shopping list

122 (0000 0000)
123 (0000 1111)
124-136 ASCII

StartTagSubject

CH 0
Values miss "shopping list" length + 2
shopping list

</subject> 137 (0000 0000)	<u>ElementSubject</u> EE subject
<body> 138 (0000 0000)	<u>ElementNote</u> SE(body) (body)
milk, honey 139 (0000 0000) 140 (0000 1101) 141- 151 ASCII	<u>StartTagBody</u> CH 0 (CH) Values miss "milk, honey" length + 2 Milk, honey
</body> 152 (0000 0000)	<u>ElementBody</u> EE 0
</note> 153 (0000 0001)	<u>ElementNote</u> EE 1 (body)
</notebook> 154 (0000 0001)	<u>ElementNotebook</u> EE 1 (note) End Document 0 but not encoded as 1:1 ED

```
<?xml version="1.0" encoding="UTF-8"?>
  <notebook date="2007-09-12">
    <note date="2007-07-23" category="EXI">
      <subject>EXI</subject>
      <body>Do not forget it!</body>
    </note>
    <note date="2007-09-12">
      <subject>shopping list</subject>
      <body>milk, honey</body>
    </note>
  </notebook>
```

Table 47. Notebook.xml Local Copy for Quick Reference

- | |
|---|
| <ol style="list-style-type: none"> 1. Decode Event Code (according to grammar-rule context) 2. Decode event content (Table 42 and Table 43) 3. Move forward in grammar according to current grammar rules 4. Return to step 1 if last event was not EndDocument (ED) 5. [Done] |
|---|

Table 48. Pseudocode Algorithm for Decoding EXI Streams

1. Strict Encoding

A final caveat unique to schema-informed processing is the option **STRICT**. When this encoding option is set, any deviation from the schema's defined grammars are considered to be fatal errors. Without the **STRICT** option set, the schema-informed encoding can adapt to undefined events (that is undefined by schema events) representing those events by reverting to the higher entropy schemaless event code encoding rules.

3. EXI to XML Schema Datatypes and Event/Content Representations

Table 49 list EXI's small set of datatype mappings that each XML document's events must be transformed into before creating an EXI stream output. Important to notice here is that value contents are not explicitly defined in this map because EXI allows the capitalization of schema datatypes as shown in Table 50. Therefore, the value content can be of any datatype defined by an XML schema, with the default type being **String**.

Event Content Item	Event	Default Datatype
name	PI, DT, ER	String
prefix	NS	String
local-element-ns	NS	Boolean
public	DT	String
qname	SE, AT	qname
system	DT	String
text	CM, PI	String
uri	NS	String
value	CH, AT	According to schema

Table 49. Default Datatype Of EXI Events (From W3C, 2008)

A few specific datatype notes pertain:

- When the `preserve.lexicalValues` option is set to false, the default setting, value content items are represented according to their schema-defined datatypes as listed in the Table 50 datatype mapping. In the absence of a schema or the `preserve.lexicalValues` option is set to true, all value content items are represented as strings, but with the values restricted as defined in Table 51 to ensure valid datatype format.
- The List datatype, an array of values, is not mapped explicitly because its datatype mapping is dependent upon the schema datatype representation of the values in the list. For example, if the list is of type integer, then the list will be encoded as a set of integers. Lists are prevalent in the scientific and simulation domains.

Each EXI datatype from Table 50 is paired to a schema datatype or datatypes as a translation from XML Schema datatypes to built-in EXI datatype representation.

EXI Datatypes	Schema Datatypes	Schema Datatype Notes
Binary	xsd:Base64Binary	base64Binary
	xsd:hexBinary	hexBinary
Boolean	xsd:boolean	boolean
Date-time	xsd:dateTime	dateTime, time, date, gYearMonth, gYear, gMonthDay, gDay, gMonth
Decimal	xsd:decimal	decimal
Float	xsd:double	float, double
n-bit Unsigned Integer	xsd:integer	<p><i>integer</i>, the representation of which depends on the facet values as follows.</p> <p>When the bounded range of <i>integer</i> is 4095 or smaller as determined by the values of minInclusive, minExclusive, maxInclusive and maxExclusive facets, use n-bit Unsigned Integer representation.</p> <p>Otherwise, when the <i>integer</i> satisfies one of the followings, use Unsigned Integer representation.</p> <ul style="list-style-type: none"> - It is <i>nonNegativeInteger</i>. - Either minInclusive facet is specified with a value equal to or greater than 0, or minExclusive facet is specified with a value equal to or greater than -1. <p>Otherwise, use Integer representation.</p>
Unsigned Integer		
Integer		
String	xsd:string	String, any simple type, any URI, all types derived by union
List		All types derived by list, including IDREFS and ENTITIES
QName		

Table 50. Schema to EXI Default Datatype Transformation Mapping
(From W3C, 2008)

EXI Datatype ID	Restricted Character Set
xsd:base64Binary	{ #x9, #xA, #xD, #x20, +, /, [0-9], =, [A-Z], [a-z] }
xsd:hexBinary	{ #x9, #xA, #xD, #x20, [0-9], [A-F], [a-f] }
xsd:Boolean	{ #x9, #xA, #xD, #x20, 0, 1, a, e, f, l, r, s, t, u }
xsd:dateTime	{ #x9, #xA, #xD, #x20, +, -, ., [0-9], :, T, Z }
xsd:decimal	{ #x9, #xA, #xD, #x20, +, -, ., [0-9] }
xsd:double	{ #x9, #xA, #xD, #x20, +, -, ., [0-9], E, F, I, N, a, e }
xsd:integer	{ #x9, #xA, #xD, #x20, +, -, [0-9] }

Table 51. EXI Built-In Datatype Character Restrictions (From W3C, 2008)

4. EXI Datatypes

This section discusses the details of EXI datatypes, presented using Java example code based on the OPENER-EXI implementation.

a. *Unsigned Integer Datatype*

Unsigned Integers are fundamental to the EXI encoding process as they are used regardless of schema versus schemaless encodings or alignment option. If byte alignment is used, all event codes and string table hit/miss values are encoded with Unsigned Integers instead of n-bit integers.

Unsigned Integers are encoded similar to that of the version field of the EXI header, except that instead of a 4-bit pattern for each integer, an 8-bit pattern is used. Again like the version field, an unsigned integer is able to support integers of arbitrary magnitude.

Unsigned Integers are represented as a sequence of octets with the sequence terminating at the octet with its most significant bit, left most, set to 0. The numerical value is kept in the 7 least significant bits, right most of the octet. Example comparison of EXI Unsigned Integer to traditional binary values are detailed in Table 52. The overall value of the unsigned integer is achieved by adding the consecutive octets after being multiplied by an

octet digit multiplier. Table 53 provides the Unsigned Integer decode pseudo code and Table 55 provides the OPENER-EXI Java based implementation of decode. Table 54 provides the Unsigned Integer encode pseudo code and Table 56 provides the OPENER-EXI Java based implementation of encode. In general, the key to the unsigned-integer process is the understanding that the least significant byte is the left most, first byte, opposite as how an integer would be written on paper.

8-Bit Integer (EXI)	Decimal Value	Normal Bit Pattern (32 Bits)
00000000	0	00000000000000000000000000000000
00000001	1	00000000000000000000000000000001
01111111	127	000000000000000000000000000011111111
10000000 00000001 (0 + 1*128)	128	0000000000000000000000000100000000
10000001 00000001 (1 + 1*128)	129	0000000000000000000000000100000001
10110001 00000111 (49 + 7*128)	945	0000000000000000000000001110110001

Table 52. Verbose EXI Unsigned Integer Value Examples

1. Start with the current value set to 0 and the initial multiplier set to 1.
2. Read the next octet.
3. Multiply the value of the unsigned number represented by the 7 least significant bits of the octet by the current multiplier and add the result to the current value.
4. Multiply the multiplier by 128.
5. If the most significant bit of the octet was 1, go back to step 2.

Table 53. EXI Unsigned Integer Decoding Pseudocode Algorithm

1. read 7 bits and format as a normal 7 bit binary number
2. if more "1" bits remain ((n >> 7) > 0) // 7 bit shift to right
 - OR the 7 bits from step 1 with 128
 - append the 8 bits to buffer
 - repeat step 1 until no more 1 bits remain

Table 54. EXI Unsigned Integer Encoding Pseudocode Algorithm


```

public static int readUnsignedInt(InputStream istream)
    throws IOException {
    int multiplier = 1;
    int value = 0;
    int valueBuff = 0;
    int intIn = 0;
    do{
        intIn = istream.read();
        valueBuff = intIn & 0x7F;
        value += valueBuff * multiplier;
        multiplier *= 128;
        intIn = intIn >> 7;
    }while(intIn > 0);
    return value;
}

```

Table 55. EXI Unsigned Integer Decoding Java Method

```

public static int howMany7BitBytes(int n){
    int howmany = 1;    // how many 7 bit bytes to write
    if (n < 0)
        throw new IllegalArgumentException("must be a positive
            number to " + "DataTypeUnSignInteger.howMany7BitBytes
            [" + n +"]");
    while((n = n >> 7) > 0) {    howmany++;    }
    return howmany;
}
public static int[] getEncodedUnsignedInt(int n){
    int howMany = howMany7BitBytes(n);
    int[] byteToEXI = new int[howMany];
    for(int i = 0; i < howMany; i++) {
        byteToEXI[i] = (int)(n & 127);
        if(i != howMany - 1)
            byteToEXI[i] += 128;
        n = n >> 7;
    }
    return byteToEXI;
}
public static void writeUnsignedInt(OutputStream outputStream, int x)
    throws IOException {
    int[] outBytes = getEncodedUnsignedInt(x);
    for(int i = 0; i < outBytes.length; i++) {
        outputStream.write(outBytes[i]);
    }
}

```

Table 56. EXI Unsigned Integer Encoding Java Method

b. n-bit Unsigned Integer Datatype

An n-bit unsigned integer is used during bit aligned EXI operations, default alignment for encoding event codes, prefix component of a QName as well as certain value content items. N-bit integers maximize compactness by using only the minimum number of bits necessary to represent an index. However, certain conditions that EXI must work within cannot use bit-aligned data:

- Digital signature
- Encryption
- EXI compression or Pre-Compression (byte alignment required)
- The byte-aligned encoding option is set

Note that when an N-bit Integer is called, but the encoding is not under an N-bit condition, an EXI Unsigned Integer is used in its place.

[illegible]

c. *String Datatype*

Strings are nothing more than a list of characters as would be in any other encoding except that an EXI string is encoded as a length-prefixed sequence of characters. The length, represented as an Unsigned Integer, indicates the number of characters in the string that follows.

If a restricted character set is defined for the string, each character is represented as an n-bit Unsigned Integer, that is, an index into the restricted character set the represent the character being encoded. The following steps listed in Table 57 are used to determine a restricted character set's values.

First, determine the character set for each datatype in the datatype hierarchy of the string value that has one or more pattern facets.
For each datatype with more than one facet <ul style="list-style-type: none"> Compute the restricted characters set based on the union of expression (patterns) If this datatype contains at least 255 characters of non-BMP characters <ul style="list-style-type: none"> The character set of the datatype is not restricted
The restricted character set for the sting is the intersection of all character sets from the previous step <ul style="list-style-type: none"> If the character set contains less than 255 <ul style="list-style-type: none"> The string is a restricted character set Each character is represented using an n-bit integer $n = \log_2 (N + 1)$ N is the number of characters in the restricted set

Table 57. EXI String Restricted Set Pseudocode Algorithm

A pictorial representation of this restricted string resolution process is contained in Figure 42 showing the restricted set of characters as input into the EXI processor.

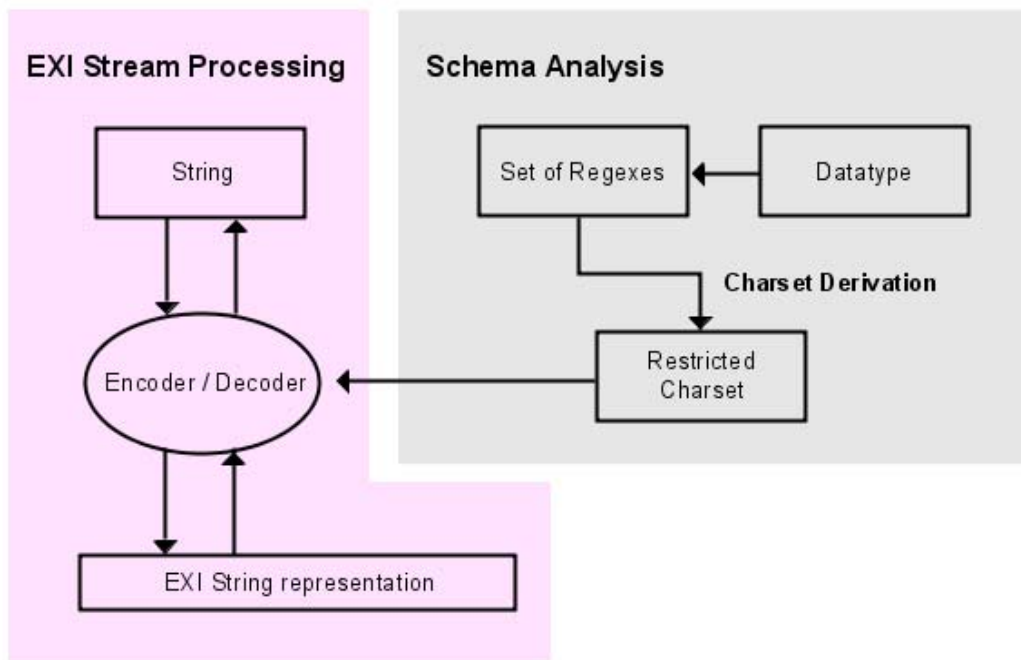


Figure 42. EXI String Processing Model (From W3C, 2008)

d. Binary Datatype

Values typed as Binary are represented as a length-prefixed sequence of octets representing the binary content, where the length is represented as an Unsigned Integer. In reality, this often how other data formats represent binary data, and EXI is no exception.

e. Boolean Datatype

There are two conditional representations for a Boolean: one with a defined pattern facet and one without.

If no pattern is defined on how a Boolean is to be represented, then the default method is to encode the Boolean as an n-bit unsigned integer using a single bit. If the single bit is set to 1, then the value is to be interpreted as true otherwise a 0 is false.

If a pattern is defined, then EXI can make lexical distinction between "0", "1", "false" and "true", which enables arithmetic operations on Booleans, and or logical strings to be retained. Again, an n-bit unsigned integer is used, but with n of the n-bit set to 2 (2 bits) so that all 4 cases can be represented: "0", "1", "false" and "true". The 4 binary values of {00 (0), 01 (1), 10 (2), 11 (3)} represents value "false", "0", "true" and "1," respectively.

f. Float Datatype

Floats are represented as two consecutive Integers. Pseudocode to decode a float is listed in Table 58.

The first Integer represents the mantissa (integer value) of the floating point number. The range of the mantissa is [-263, 263-1]. The second Integer represents the base-10 exponent of the floating point number. The range of the exponent is [-214-1, 214-1].

Note that the exponent value -214 is used to indicate special values: infinity, negative infinity and not-a-number (NaN). An exponent value -214 with mantissa values 1 and -1 represents positive infinity (INF) and negative infinity (-INF) respectively. An exponent value -214 with any other mantissa value represents NaN. Values typed as Float with a mantissa or exponent outside the accepted range are represented as schema-invalid values. If this case arises, a possible work around is to use the EXI Decimal datatype. Initial evaluation of the EXI specification leads to the belief that the Decimal datatype, defined next, to be a more robust and flexible representation of non-integer values.

1. Retrieve the mantissa value m as an integer
2. Retrieve the exponent value e as an integer
3. If the exponent value is -(214), <ul style="list-style-type: none"> the mantissa value 1 represents INF, the mantissa value -1 represents -INF any other mantissa value represents NaN.
4. If the exponent value is not -(214), <ul style="list-style-type: none"> m is the mantissa e is the exponent obtained in the preceding steps. the float value is $m \times 10^e$

Table 58. EXI Float Decoding Pseudocode Algorithm

g. Decimal Datatype

Decimals are unique to floats in that Decimals can represent values of arbitrary magnitude and precision as there is no upper or lower bounds on their values. Decimals may be a more robust datatype than Float depending on domain needs.

Decimals are represented as a Boolean sign followed by two Unsigned Integers.

A sign value of zero (0) is a positive value and one (1) is a negative value.

The first Unsigned Integer represents the integer portion of the Decimal value, non-fractional part of the number such as the 43 from the decimal 43.68.

The second Unsigned Integer represents the fractional portion of the Decimal value with the digits in reverse order to preserve leading zeros, that is, the fractional part counts up from the right to the left in terms of bits. This is no different from any other binary encoding of decimals, but is not how humans read decimals. For the example 43.68, the second Unsigned Integer is 68 and not 100/68 as the real value would be. For the number 12.125, the second Unsigned Integer would be 125. Table 59 defines a pseudocode algorithm for Decimal encoding.

```

Decimal d
SignFlag = is d < 0
FirstUnsignedInteger = (integer cast)d
d = |d| - FirstUnsignedInteger
SecondUnsignedInteger = 0
multiplier = 1
int nextVal = 0
while(d > 0){
    nextVal =d <<1
    d = d << 1
    SecondUnsignedInteger += nextVal * multiplier
    Multiplier *= 10
}

```

Table 59. EXI Decimal Encoding Pseudocode Algorithm

h. Integer Datatype

The Integer type supports signed integer numbers of arbitrary magnitude. Values typed as Integer are represented as a Boolean sign followed by an Unsigned Integer.

A sign value of zero (0) is used to represent positive integers and a sign value of one (1) is used to represent negative integers.

For nonnegative values, the Unsigned Integer holds the magnitude of the value. For negative values, the Unsigned Integer holds the magnitude of the value minus 1 as the negative-numbers portion also represents the value 0.

i. QName Datatype

QNames represent the local-name and prefix of XML elements and attributes. Prefix and local-name components are encoded as strings, though if the preserve.prefixes option is set to false prefixes will be discarded. If a QName is in the no namespace URI, the URI is represented by a zero-length string. Table 60 defines the rules for QName coding

1. If the prefix is defined, select the m-th prefix value associated with the URI of the QName as the candidate prefix value. Otherwise, there is no candidate prefix value.
2. If the QName value is part of an SE event followed by an associated NS event with a local-element-ns flag value being true, the prefix value is the prefix of such NS event. Otherwise, the prefix value is the candidate value, if any, selected in step 1 above.

Table 60. EXI QName Rules

j. Date-Time Datatype

Date and time datatypes are encoded as a sequence of values representing their individual parts (hour, day, year, ...). Note that the order of date and time components are not directly specified as they are dependent on the Schema associated with the XML document. Therefore, military dates in the format of DD-MM-YY are as valid as the more common civilian method of MMM-DD-YYYY. The breakdowns of time and date components are listed in Table 61 and Table 62. For Table 62, square bracketed items are included if and only if the value of its preceding presence indicator is set to true.

Date/Time Components	Valid Component Values	EXI Component Datatype
Year	Offset from 2000	Integer
MonthDay	Month * 32 + day	9-bit Unsigned Integer where day is a value in the range 1-31 and month is a value in the range 1-12.
Time	((Hour * 60) + Minutes) * 60 + seconds	17-bit Unsigned Integer
FractionalSecs	Fractional seconds	Unsigned Integer representing the fractional part of the seconds with digits in reverse order to preserve leading zeros
TimeZone	TZHours * 60 + TZMinutes	11-bit Unsigned Integer representing a signed integer offset by 840 (14 * 60)
presence	Boolean presence indicator	Boolean

Table 61. Components Fields of a Date and Time Datatype (From W3C, 2008)

XML Schema Type	EXI Date And Time Component Mapping
gYear	Year, presence, [TimeZone]
gYearMonth	Year, MonthDay, presence, [TimeZone]
date	
dateTime	Year, MonthDay, Time, presence, [FractionalSecs], presence, [TimeZone]
gMonth	MonthDay, presence, [TimeZone]
gMonthDay	
gDay	
Time	Time, presence, [FractionalSecs], presence, [TimeZone]

Table 62. Schema To EXI Date and Time Component Mapping (From W3C, 2008)

k. Enumerated Datatype

Schemas can provide one or more enumerated values for types. EXI exploits those pre defined values when they are available to represent values of such types in a more efficient manner than it would otherwise if using built-in EXI datatypes. Values of enumerated types are encoded as n-bit Unsigned Integers where $n = \lceil \log_2 m \rceil$ and m is the number of items in the enumerated type. The value assigned to each item corresponds to the ordinal position of the enumeration in schema-order starting with position zero. Exceptions are for schema types derived from others by union and their subtypes, QName or Notations, and types derived by restriction. The values of such types are processed by their respective built-in EXI datatype representations instead of being represented as enumerations.

Enumeration indices add a degree of complexity, but achieve higher compactness than using raw enumeration values.

l. List Datatype

Lists are common within scientific and simulation XML documents, and require special coding in order to be consistent with previously described formats. Lists are first encoded with their length then the number of items in the list as an unsigned integer is then followed by its contents encoded in accordance with their schema-defined datatype.

m. Multiple Ancestor Datatype

If a schema datatype is derived from multiple datatypes, the closest ancestor schema datatype is how the EXI datatype is to be encoded. For example, the `xsd:int` datatype is derived from the `xsd:integer` which itself is derived from the `xsd:decimal` datatype. Because `xsd:integer` is the closest ancestor, an `xsd:int` datatype will be encoded as an EXI Integer given `xsd:integer` is mapped to EXI Integer in accordance with Table 50.

While this example may be obvious, as datatypes become more complex, implementers must be diligent to ensure proper mappings at each level of complexity.

5. Datatype Representation Map

Normally the default mappings of schema datatype to EXI datatype as listed in Table 50 are assumed, but whenever that is not the desired case, there are two methods to define alternative datatype mappings.

a. Primary Alternative Datatype Mapping

The preferred method is to define the alternatives by annotating the alternative within the EXI header Options part 4 of 5. This annotation is done by means of a XML fragment that lists the schema-defined datatype and the desired alternative datatype mapping to be used by the EXI processor. Table 63 provides an example of an XML fragment datatype representation map. The datatype representation map XML

fragment consists of a single element named “datatypeRepresentationMap” for each of the alternative mappings. Within this element, two other elements are contained:

- The first sub element is the schema defined-datatype that maps to other than a default type, i.e., the datatype to be overridden.
- The second sub element is the schema datatype to be used for the default Schema to EXI mappings, i.e., the casted type.

This approach is the preferred mapping method whenever the defaults are not to be used because it ensures interoperability.

```
<!--  
alternative mappings of schema decimals to schema string  
for EXI processing instead of using the default EXI decimal  
mapped type, e.g., decimal  
-->  
  
<datatypeRepresentationMap  
xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
  <!-- Datatype that is to be overridden -->  
  <xsd:decimal/>  
  <!-- Datatype that the previous is cast to -->  
  <xsd:string/>  
</datatypeRepresentationMap>
```

Table 63. Example EXI DatatypeRepresentationMap XML Document
(After W3C, 2008)

b. Secondary Alternative Datatype Mapping

The second method of defining alternative mappings is to use an implementation-specific mapping; one for which both the encoder and decoder have a preexisting agreement. This method does not use the header field to define the mapping, making interoperability between arbitrary systems questionable. This method is usually only suitable for production systems that process consistent data and do not need the explicit datatype mapping to understand the documents or streams, hopefully also delivering a slightly more compact format.

c. Error Reporting for Alternative Datatype Mapping

Errors are thrown only if an EXI processor tries to decode a datatype it does not know. Aside from outright unknown datatypes, which should not occur given the default all-encompassing string type, a case where this may occur is incorrectly typed data such as string facets.

An example of this error would be if an element was typed with a string facet only defining lowercase letters between “a” and “f”, but the XML document contained a date in the form of “apr 15 2009”. The EXI processor knows strings so it does not initially throw an error. The initial letter of the date ‘a’ would pass the string facet’s expectations, but the second letter ‘p’ is out of range and would cause the EXI processor to report an error.

```
<xs:pattern value="[a-f]"/>  
  
<elementName data="apr 15 2009"/>
```

The EXI processor has no way of knowing the XML date input is the incorrect datatype before processing and finding the out of range string facet, and because EXI knows the string datatype, it continues processing until the string exception is discovered.

d. Datatype Mapping Conditional Reporting

The EXI specification defines a set of reporting policies for an EXI processor that tries to decode an EXI stream that contains user-defined datatypes that also contains errors:

- May report a WARNING if the user-defined datatype in the header is not recognized.
- Must report an ERROR when an EXI processor encounters a typed value that was encoded by a user-defined datatype that it does not support.

6. Datatype Compression

In addition to EXI's inherently compact XML representation, additional compactness is obtainable with an inline compression step. If the EXI output alignment of compression is selected, as events are received from the XML document, they are not immediately written to the output EXI stream, but rather are channelized, grouped, and then compressed with a traditional text-based compression technique. Inline compression refers to the fact that the EXI stream is compressed as the events occur, and not after the entire XML document has been converted to EXI. Inline compression is used instead of post-EXI processing because the output of post-EXI processing file is in binary format, lacking the necessary redundancies required for most compression techniques to work. EXI's compression algorithm takes the already compact EXI stream of the original XML document and compresses it with the DEFLATE compressed data format and decompressed with the INFLATE format defined by RFC 1950 and 1951 to deliver a more compact EXI stream. The Java Standard Edition library package `java.util.zip` contains classes to perform both INFLATE and DEFLATE.

There are a few caveats that must be adhered to before EXI Compression or the precompression alignments can be used:

- Precompression or Compression must be set true in the header options.
- Precompression performs all steps up to the actual DEFLATE algorithm, and then writes each blocked-channeled set of events directly to the output EXI stream.
- Compression performs all steps as well as DEFLATE compression algorithm on the blocked-channeled-compressed set of events.
- Both Precompression and compression use byte alignment. Byte alignment is needed because text-based compression techniques work on redundant bytes of character data, and not raw binary bits.

a. Blocking the Stream of EXI Events into Bins

The first step is the EXI stream of events are divided into blocks, i.e., non-overlapping bins of events. All blocks will have the same number of events as defined in the EXI header blockSize. However, the last block of the EXI stream may be smaller than the others due to file size and block size settings. EXI compression splits the EXI stream into equally sized, bins of data followed by the last block

A block size is a window of visibility for the compression algorithm. Referring back to the Background and Related Work chapter, text-based compression works on redundant data within the scope of a window. The window capitalizes on the likelihood of data relatively close together will have more commonality than data further apart. This thesis, for example, is broken into chapters and sections with each having a particular focus and commonality. This section is addressing EXI compression, but chapters prior did not address compression and would have little redundant information compared to this section. A window sets the range to find redundant information, tokens, while keeping the total number of tokens small. The fewer the number of tokens, modified by the \log_2 ceiling, the more compactly the information can be indexed.

Adjusting the blockSize for a particular family of XML documents may achieve higher levels of compression by taking advantage of that families structural characteristics.

b. Channelizing the Blocks of Events

Each block is then multiplexed into channels: one structure channel and multiple distinct-value channels. The structure channel is the first and longest, containing the stream-ordered list of events and their content, excluding AT and CH events values. Each distinct AT and CH event becomes its own value channel, corresponding to the distinct element and attribute events within the block. Each value channel, like the structure channel, is in stream order, containing only values for its particular qname.

Note, xsi:type and xsi:nil attributes are stored in the structure channel and not a dedicated value channel because they are general XML document-structure characteristics and not content contributing.

Figure 43 depicts the process of blocking and channelizing an EXI stream. The example stream consists of 2047 events made from four distinct attribute and element events: two Attributes and two elements. The structure channel has 2047 entries, one entry for each of the 2047 events, and four value channels that each have varying number of entries relative to the block. Reviewing the figure, it can be seen that AT(B) generated value 1 (v1) and value 5 (v5), and each of these values are found within Channel B. The same process is repeated for all the other value channels. Important to notice here is the SE(A) is the first EXI event, but the last channel. The SE(A) contents are not processed until after the associated attributes and in this case its subelements. EXI event processing order is start-element, attributes, and then element content, which can be other elements and or content values.

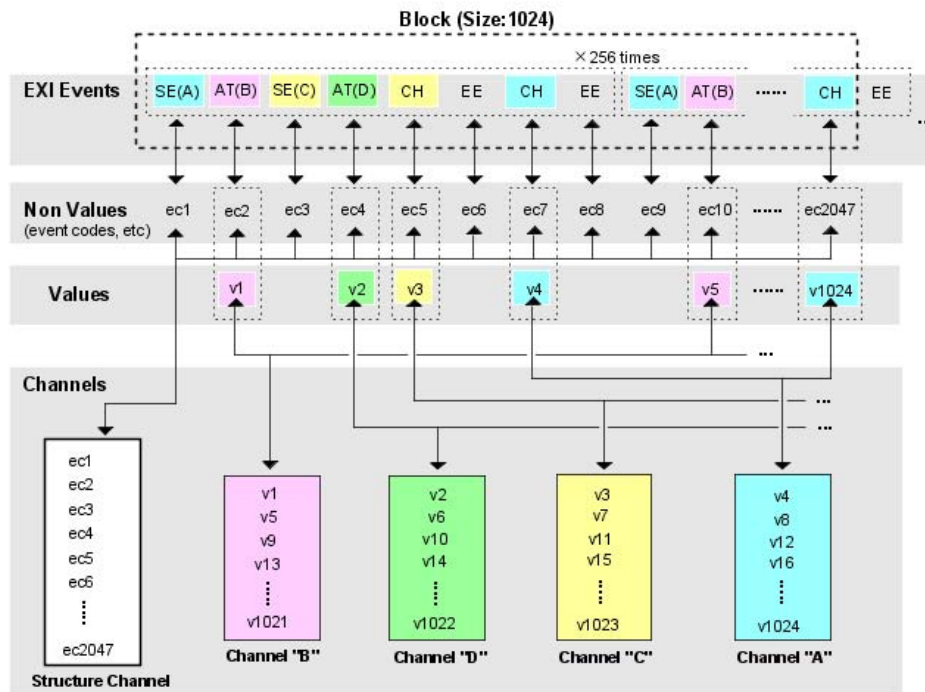


Figure 43. EXI Compressed Aligned Output Events Mapping to Compression Channels (From W3C, 2008)

c. Compressing the Channels into Streams

After a block has been channelized, it is then organized into compressed streams. The channel organization process groups smaller channels together to make a single large channel, though it does not split larger channels. The goal is to have one or more channels, each approximately the same size. The rules for combining channels:

1. If a block contains 100 or fewer values:
 - The block will contain only 1 compressed stream: structure channel followed by all of the value channels.
 - The order of the value channels within the compressed stream is defined by the order in which the first value in each channel occurs in the EXI event sequence, document order.
2. If any block contains greater than 100 values:
 - The first compressed stream contains only the structure channel.
 - The second compressed stream contains all value channels that contain no more than 100 values.
 - The remaining compressed streams each contain only one channel, each having more than 100 values.
 - The order of the value channels, second and subsequent streams, are defined by the order in which the first value in each channel occurs in the EXI event sequence.

The streams are written to the output EXI stream using the DEFLATE algorithm, and read in from the EXI stream using the INFLATE algorithm, both easily implemented with the Java Standard Edition built in classes found within the `java.util.zip` package.

Figure 44 provides a pictorial example of event blocks being channelized and compressed. The thick black lines are the structure channels and each colored line is

a distinct value channel. This figure also shows the reverse decoding as the structure and values channels are stored in EXI stream order, the reversing of the process is relatively straightforward.

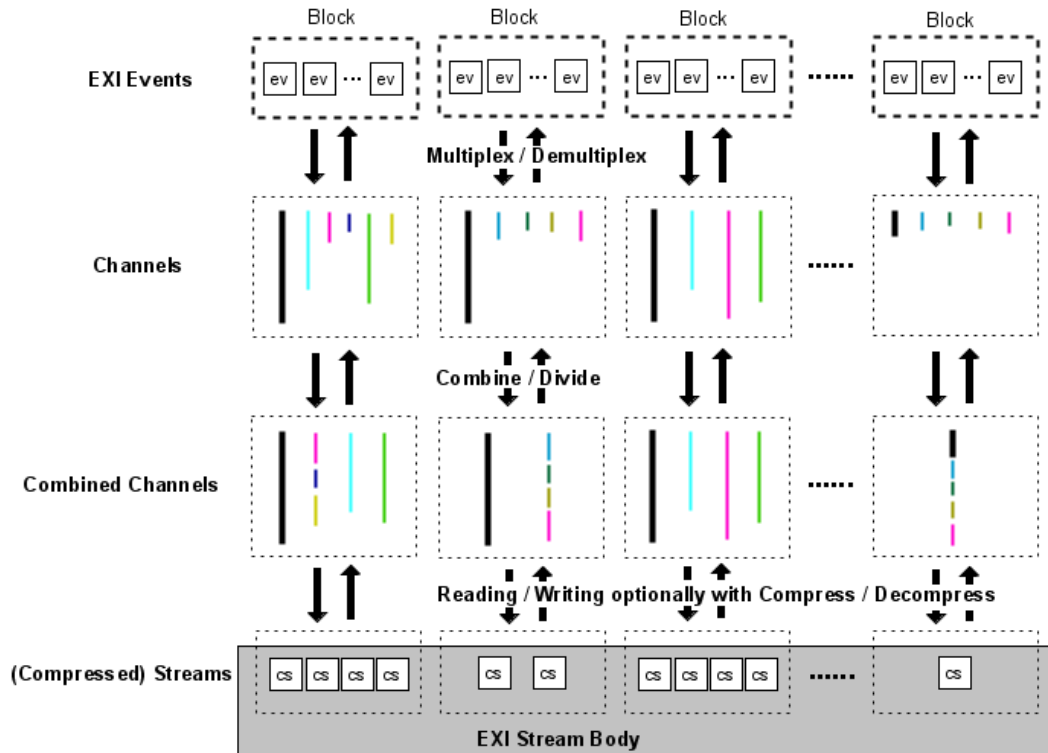


Figure 44. EXI Events To Compressed Stream for EXI Compression Aligned Output (From W3C, 2008)

It needs to be highlighted that the compression operations change the order of the events compared to the XML document. Care must be exercised to properly order both the structure and value channels to ensure that the constructs of EXI function correctly by applying the correct values and structure, namely grammars and string tables.

d. EXI Compression Summary

The EXI compression method splits the EXI sequence of events into a number of blocks which are then channelized on attribute and element values based on event count. The channels are then compressed (DEFLATE) or decompressed (INFLATE) to and from the EXI stream.

E. OPENER-EXI XML TEST CASES

A set of small test XML documents were created that each exercise a specific aspects of the EXI technique to demonstrate OPENER-EXI encoding compliance. Each of these test XML documents were created to be as small as possible in order to reasonably review their EXI output in byte-aligned output, while at the same time exercise the algorithm. The following subsection lists and describe the XML test documents.

1. notebook.xml - Hello World

XML document notebook.xml is the unofficial “Hello World” example code for EXI. This document test string table value hits in both local and global table entries.

For example, the category=”EXI” attribute under the first <note> element creates a global entry that the <subject> EXI value later in the document also hits. The date=”2007-09-12” attribute of the <notebook> root element creates both global and local entries. Further the second <note> element’s date attribute’s value is a local hit from the root element’s date attribute. It is also important to mention that the date attribute values are not out of scope for the second <note> because the first date attribute is declared in the root element, which gives it a global scope.

```
<?xml version="1.0" encoding="UTF-8"?>
<notebook date="2007-09-12">
  <note date="2007-07-23" category="EXI">
    <subject>EXI</subject>
    <body>Do not forget it!</body>
  </note>
  <note date="2007-09-12">
    <subject>shopping list</subject>
    <body>milk, honey</body>
  </note>
</notebook>
```

2. namespace.xml - Namespace Pruning

This document is the second version of EXI Hello World as defined in the specification support documents. It contains a single namespace declaration to exercise the pruning of namespace events. The key point is that in the decoding of the namespace if pruned, a dummy namespace prefix needs to be created. A simple method to manage this is to use an incrementing prefix place holder such as ns#, where # is the count of namespaces currently in the document. However, any method that generates a unique prefix for each namespace after pruning can also work. This document also contains comments which can simply be removed to solely test the namespace pruning without comment interference, otherwise comment pruning can also be exercised.

```
<?xml version="1.0" ?>
<!-- Sample XML Document -->
<ns:personnel xmlns:ns="urn:foo">
  <ns:person id="Boss">
    <ns:name>
      <ns:family>Smith</ns:family>
      <ns:given>Bill</ns:given>
    </ns:name>
    <ns:e-mail>smith@foo.com</ns:e-mail>
    <ns:yearsOfService>20</ns:yearsOfService>
    <ns:birthday>1955-03-24</ns:birthday>
  </ns:person>
  <!-- A second comment in this Sample XML Document -->
  <ns:person id="worker">
    <ns:name>
      <ns:family>Jones</ns:family>
      <ns:given>Bill</ns:given>
    </ns:name>
    <ns:e-mail>jones@foo.com</ns:e-mail>
    <ns:yearsOfService>5</ns:yearsOfService>
    <ns:birthday>1968-07-16</ns:birthday>
  </ns:person>
  <ns:person id="worker">
    <ns:name>
      <ns:family>Jones</ns:family>
      <ns:given>Sam</ns:given>
    </ns:name>
    <ns:e-mail>sjones@foo.com</ns:e-mail>
    <ns:yearsOfService>5</ns:yearsOfService>
    <ns:birthday>1959-01-26</ns:birthday>
  </ns:person>
</ns:personnel>
```

3. comment.xml - Comment Pruning

This is a simple document with multiple comments at different levels of the document. Comments are in the start document content level as well as the element content level of the document. If currently within the start tag content of a grammar, before the comment can be processed, the grammar must transition to its element content portion of the grammar. Also, multi-lined comments fire multiple comment events, one for each line of the comment when using the Java SAX parser.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Start document Comment on 2 lines...
2 comment events...one for each line -->
<document>
  <author>Janet Schmitt</author>
  <!-- Element level comment -->
  <date>8/17/2007</date>
  <title>
    <htmlimg src="tnation.jpg" alt="Tour Nation"/>
  </title>
  <subtitle>Bike Models</subtitle>
</document>
```

4. pi.xml - Processing Instruction (PI) Pruning

This document contains a single processing instruction, *<?xml-stylesheet type="text/css" href="bike.css" ?>*. The Target portion of the processing instruction is *xml-stylesheet* and the data is all the remains to the right of the target, *type="text/css" href="bike.css"*, though not including the *?>*. The data portion of a processing instruction can contain one or more attribute-like sets, as in this case a type and href; the data portion is not tokenized by white space.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="bike.css" ?>
<document>
  <author>Janet Schmitt</author>
  <date>8/17/2007</date>
  <title>
    <htmlimg src="tnation.jpg" alt="Tour Nation"/>
  </title>
  <subtitle>Bike Models</subtitle>
</document>
```

5. customer.xml - String Table Values Scope

This document flexes string table values scope limits. For example, the “cust201” value for the <customer> element attribute custID is placed in both the global table as well the local table for custID. When any of the <order> elements orderBy attributes looks up value “cust201,” each results in a global hit, but not a local hit due to value scope.

The first <order> element and its attributes values have not been defined prior to this declaration, and so, at best, can achieve a global hit. The second <order> element and its attributes values also have global hits and not local hits. The reason is the first <order> element lost scope at its end-element, losing reference to all values it defined. When the second <order> element performs string look-ups, it finds none and defers to global hits.

```
<?xml version="1.0" encoding="UTF-8"?>
<customers>
  <customer custID="cust201" custType="home">
    <orders>
      <order orderID="or10311" orderBy="cust201">
        <orderDate>8/1/2008</orderDate>
      </order>
      <order orderID="or10311" orderBy="cust201">
        <orderDate>8/1/2008</orderDate>
      </order>
    </orders>
  </customer>
</customers>
```

6. **dup.xml - Grammar Transitions on Duplicate Elements/Attributes**

This document test immediately repeating elements to ensure the grammars are correctly being pushed and popped off the grammar stack. The orders of the attributes are altered within the element tags to ensure proper table indexing. Additionally, new attributes are added to the last two <meta> elements to prove the continued learning after the first occurrence of an element.

```
<?xml version="1.0" encoding="UTF-8"?>
<X3D>
  <head>
    <meta content='content1' name='name1' />
    <meta content=' content 2' name='name2' />
    <meta content=' content 3' name='name3'
          nether="some" />
    <meta gain="one" content=' content 3' name='name3'
          nether="some" />
  </head>
</X3D>
```

7. **nestImmediate.xml - Grammar Transitions on Nested Elements with Same Name**

This document tests the grammar's transitions ability to handle nested same-name elements. Specifically, in this case, a <Transform> element is immediately followed by another <Transform> element before the first is terminated.

```
<?xml version="1.0" encoding="UTF-8"?>
<X3D>
  <Transform rotation='1 0 0 0.4'>
    <Transform rotation='0 0 1 0.5' />
  </Transform>
</X3D>
```

8. **fullFlex.xml - All Pruning Options**

This document flexes the entire EXI protocol with multiple namespaces, comments at multiple levels within the document, multiple processing instructions, repeating elements, attributes and values, and a fairly long, but reasonably manageable document size.

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="parts.css" ?>
<?xml-stylesheet type="text/css" href="model.css" ?>
<!-- this is a comment -->
<mod:model id="pr205"
  xmlns:mod="https://jacksonselect.com/models"
  xmlns:pa="https://jacksonselect.com/parts">
  <mod:title>Laser4C (PR205)</mod:title>
  <mod:description>Entry level color laser
    printer</mod:description>
  <mod:type>color laser</mod:type>
  <mod:ordered>320</mod:ordered>
  <mod:parts list="chx201,fa100-5,eng005-2,cbx-
    450V4,tn01-53"/>
  <!-- this is another comment -->
  <pa:parts>
    <pa:part id="chx201">
      <pa:title>Chassis and Roller
        Kit</pa:title>
      <pa:description>PR205 chassis and
        rollers</pa:description>
      <pa:instock>512</pa:instock>
    </pa:part>
    <!-- a pa:parts comment -->
    <pa:part id="fa100-5">
      <pa:title>Fuser Assembly</pa:title>
      <pa:description>Fuser assembly/JE
        series</pa:description>
      <pa:instock>1253</pa:instock>
    </pa:part>
    <pa:part id="eng005-2">
      <pa:title>Engine Kit</pa:title>
      <pa:description>Printer engine kit/JE
        series</pa:description>
      <pa:instock>3895</pa:instock>
    </pa:part>
    <pa:part id="cbx-450V4">
      <pa:title>Controller board</pa:title>
      <pa:description>PR205 printer controller
        board</pa:description>
      <pa:instock>483</pa:instock>
    </pa:part>
    <pa:part id="tn01-53">
      <pa:title>Toner Kit</pa:title>
      <pa:description>PR205 toner kit
        (b,m,c,y)</pa:description>
      <pa:instock>812</pa:instock>
    </pa:part>
  </pa:parts>
</mod:model>

```

F. SOFTWARE ENGINEERING PRACTICES EMPLOYED

1. Unit Testing

To achieve a degree of freedom in software code reliability, unit tests are used to verify methods by means of the JUnit package. JUnit is a testing framework used to implement unit tests in Java (JUnit, 2009). Using JUnit provides a well-formatted and industry-accepted method of verifying expected performance of code units verifying that the source-code methods execute as advertised.

JUnit assert tests are conducted at method headers and at return result checkpoints. For example, if an argument of a method is supposed to be a file, and not only a file, but an existing file, a JUnit test is conducted to prove the assumption is true, that the argument is an existing file, otherwise the test terminates the program with a customized error message.

Using localized custom error messages helps to quickly troubleshoot the precise point of an error, and ultimately cutting down on exceptions making a more robust source code base knowing the expected flow of the program is valid.

2. Linear Progression

To the highest extent possible, a linear program flow is used. The key is keeping the branching into and out of methods low so that program flow can be easily mapped, which results in code that is easier to understand. This enabled faster troubleshooting when errors arise because the source of the error is clearer due to the narrow scope of each method and not dependent on a list of possible branches.

When branching is unavoidable, a new method is created for each branch, and then a super method that only, in order of execution flow, calls the individual branches. This super method contains no procedural code other than the calls to submethods. This super method is where many of the unit tests are conducted as it is a central point of program flow.

3. Variable Naming

Jeff Weekley, an Audio/Video programmer in MOVES, has a simple but profound rule of thumb good for variable names: “If no one is asking what the meaning of the variable name is any more, then the variable name is good.”

Since OPENER-EXI is an implementation of the proposed EXI recommendation, which has defined events, objects and other nouns describing the specification, these specification-defined names are used to name the code objects in order to remove any ambiguity of code object intent. Additionally, traditional Object Orientated Programming (OOP) naming conventions were used:

- All classes start with an upper camel case lettering
- All methods and variables, other than constants, start with a lower camel case lettering

4. Object Oriented Piecewise Methods

Using an OOP mindset, common functionality is grouped into a single class whenever possible and extended by concrete classes for custom extensions of the base class. For example, string table parameters, string table hit no hit, indexes, and others for each event is stored in a common abstract class called EventStringTableParameters instead of being individually implemented within each concrete event class: EventStartElement, EventAttribute, EventComment to name a few. Each concrete event class then extends the EventStringTableParameters class to gain access to every event common string table parameters function consistently between concrete classes.

Placing all common methods in a super class instead of individually within each concrete class cuts down on number of lines of code, and makes troubleshooting and revisions easier. Changes are only made one time within a single super class, which then automatically cascades those changes to all other implementing concrete event classes instead of performing many repeated exact duplications of the same code change within each concrete class.

G. W3C STATUS OF EXI RECOMMENDATION

As of this thesis, the EXI specification is in Candidate Recommendation (CR) status, the second of four steps in the W3C specification recommendation process. At this phase of development, multiple independent implementations of EXI are being developed and tested for interoperability. The purpose of this is to verify the specification is written well enough that developers are able to understand the specification to implement it (W3C, 2005). The details of the interoperability work plan are accessible only by EXI members. However, the current in progress work is in building a robust testing framework to prove the implementations are interoperable.

Upon graduation from CR, the EXI specification will move to Proposed Recommendation (PR) where it is community reviewed. The intent of this phase is to ensure conformance with existing practices and that nothing was missed in the previous phases (W3C, 2005). This phase validates that EXI is what is says by people other than those who have been deeply involved with the evolution of EXI; independent outside inspection of the specification.

From PR, if approved by the W3C EXI advances to Recommendation (REC) and receives W3C's recommendation for wide adoption of the specification (W3C, 2010). At this point EXI is a widely accepted standard.

Why is all this effort needed? Such due diligence is necessary to ensure the standard is well vetted by the industry so that it does not degrade the existing body of standards. This provides any implementer assurance of interoperability and long term IT industry growth.

When will EXI receive W3C Recommendation? There is no specified time as it is dependent upon the successful completion of the recommendation and preceding phases. However, the internal EXI working optimistically hopes to go for recommendation by the end of 2010.

H. CHAPTER SUMMARY

This chapter covered the W3C EXI specification and describes many of the core concepts with detailed code examples from the NPS OPENER-EXI implementation. The EXI header fields are defined in terms of format as well as the algorithms used to generate the header-specific fields. The methods employed by EXI to process an XML document are discussed. EXI event codes are defined as well as grammar event ordering and transitions. The EXI binary datatype mappings are listed with both pseudocode and Java source code demonstrating the procedure to transform XML events to binary EXI events. The optional EXI compression alignment is defined and depicted. The collection of XML documents used to test the OPENER-EXI implementation is defined and annotated with the corresponding subsets of the EXI specification that each document exercises. The chapter concludes with the software engineering practices used in the creation of OPENER-EXI.

IX. DEMONSTRATION AND ANALYSIS OF RESULTS

A. INTRODUCTION

This chapter conducts an experiment of the EXI technique compared to other DoD compression on DoD-relevant XML documents, and provides analysis of the results. A generalized XML testing repository of XML documents is defined. Recommended EXI settings specific for the DoD domain are listed with justifications. Statistical measures are presented that prove the superior effectiveness of EXI compared to the other DoD compression techniques. A set of generalized predictive models of the EXI technique are also presented to enable prospective EXI implementers and users a degree of predictability regarding the expected effect of EXI when applied to their XML domain. The chapter ends with a set of downloadable development tools that exercise the EXI specification.

B. DOD-SPECIFIC EXI EXPERIMENTAL TEST CASES DEFINED

To show the effect of the EXI technique in support of the DoD's Network-Centric data-sharing vision, a collection of three broad categories of DoD-focused XML documents are defined and exercised with EXI. The categories exercised are centered on DoD M&S, General DoD such as office automation and logistics, and representative tactical DoD messages.

The focus of this experiment is bandwidth savings to enable existing network devices to receive more information without altering the established network architecture. The primary concept is that the more compressed a file becomes, the less bandwidth is required to transfer it. For example, if a file can be compressed to half its size, then a theoretical doubling of bandwidth potential can be realized. That is, two files might be transferrable under the same conditions as the original.

1. DoD Modeling and Simulation Sub-Category Test Cases

The following descriptions define the XML documents used for the DoD Modeling and Simulation category.

Haar: The OpenCV artificial intelligence vision package uses XML to configure its real-time Haar cascading facial detection algorithm (Bradski, 2008). The HAAR facial-detection technique is an efficient classification method that uses facial features from a large training repository of faces to determine what a face looks like in order to classify tactical images as having a face or not. This is accomplished by solving the eigenvectors of key features from a training set that generalize what a face is. The cascading portion of the technique denotes that multiple classifiers are used in the classification process. An example of the HAAR facial recognition results is shown in Figure 45. Using XML enables developers the ability to rapidly alter the HAAR configuration to meet unique environmental conditions without requiring direct C code interaction and recompilation.

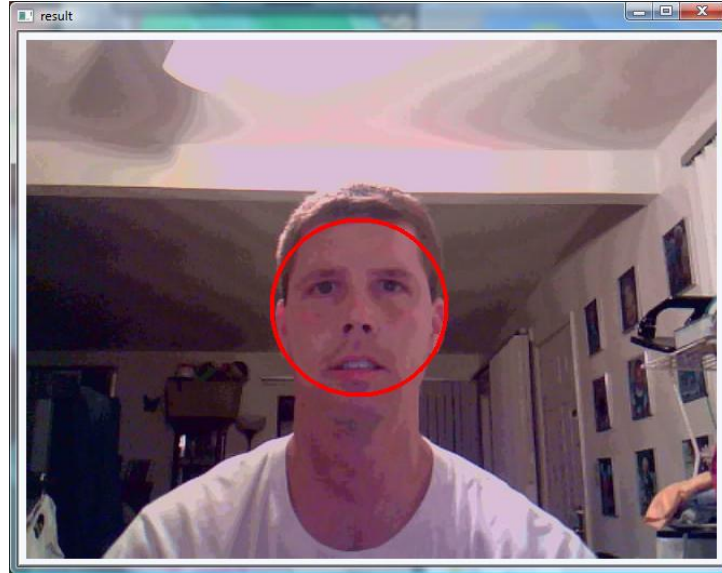


Figure 45. Open CV HAAR Facial Detection Example Results

Scalable vector graphics (SVG): SVG an XML language for defining 2D graphics used for Web applications and mobile devices. The SVG file in Figure 46 is from the open-source SVG editor Inkscape (Inkscape, n.d.).



Figure 46. Example Rendered SVG File (From Inkscape, n.d.)

OOB: Orders of Battle (OOB) define or list what entities, along with attributes about the entity, are available within a simulation scenario. XML-based OOBs enable heterogeneous simulators to use a common battle-start configuration file.

Military Scenario Definition Language (MSDL): MSDL a standardized XML format for describing the state of military action within DoD military simulations.

Autonomous Vehicle Control Language (AVCL): AVCL an XML languages used in the command and control of autonomous unmanned vehicles to represent mission(s): planning, scripting, and replay.

Discrete Event Simulation (DES): The DES tool VISKIT, based on SIMKIT, uses XML as the configuration and storage medium for its models (VISKIT, 2008; SIMKIT, 2009; Buss, 2001; Buss, 2002). Figure 47 and Figure 48 are VISKIT's visual representation of the classic M/M/1 server queue model. The M/M/1 queue model represents, in Kendal notation, a single server that has arrivals and provides services using at Random Variate M time (Ross, 2007).

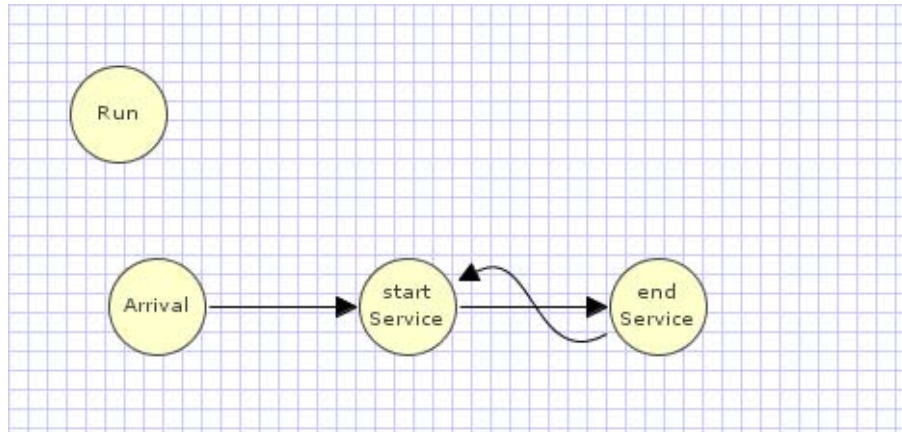


Figure 47. VISKIT/SIMKIT Example M/M/1 Queue Event Graph

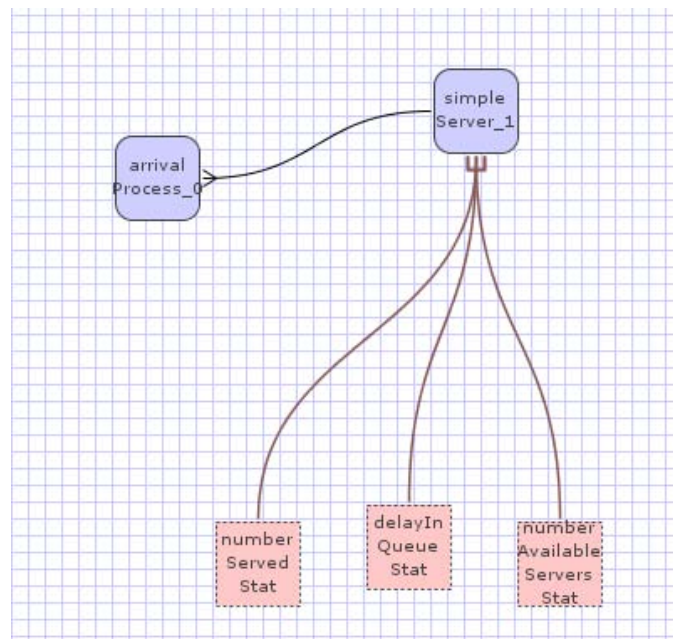


Figure 48. VISKIT/SIMKIT Example M/M/1 Queue Assembly

X3D: Is a standardized XML language for defining 3D computer graphics displayable in Web and other browsers. Figure 49 is a simple example X3D scene showing all five platonic solids.

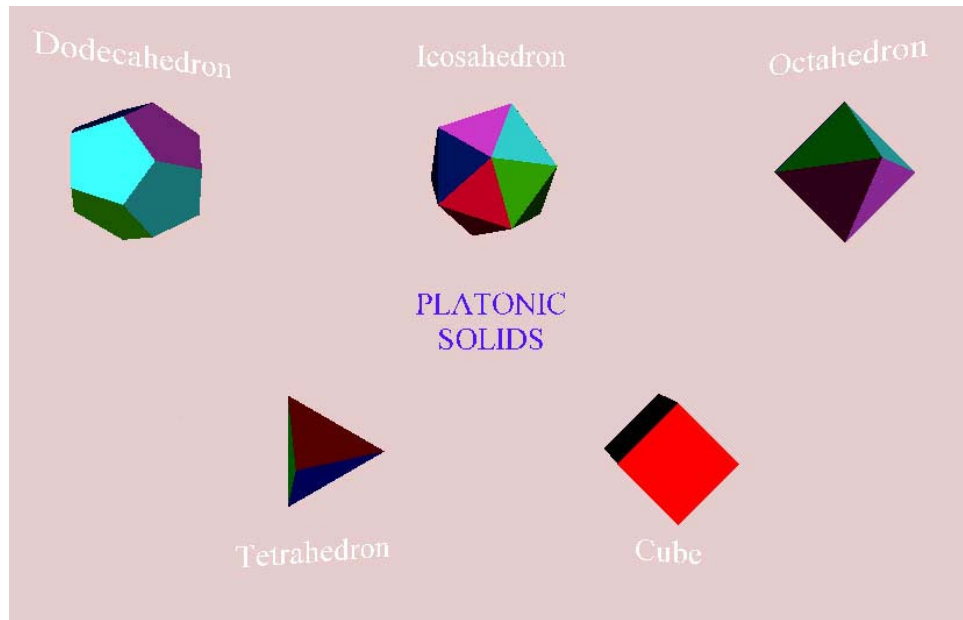


Figure 49. Example X3D 3D Scene of the 5 Platonic Solids

Humanoid: The open source Delta3D gaming and simulation engine uses XML for its character maps to define skeletal structure, texture mapping, motion model, and other human behaviors and attributes (Delta3D, n.d.). Figure 50 is a Delta3D example with a Marine uniform texture.



Figure 50. Example Delta3D Humaniod Map

3D Map: Delta3D also uses XML for its world mapping to define the virtual world terrain, the objects that are in the virtual world, the physics of the virtual world, and other attributes. Figure 51 is an example of a war-ridden desert town.



Figure 51. Example Delta3D Scene Map

Distributed Interactive Simulation (DIS): DIS is a binary IEEE standard format that enables distributed interactive communicate of M&S object interactions (DIS, 1995). The Open-DIS project also represents DIS data with a XML format (Open-DIS, n.d.).

2. DoD General Sub-Category Test Cases

The following descriptions define the XML documents used for the DoD General category, applicable to standard daily operations.

XHTML: An XML language for representing Web content within Web browsers. It is similar to HTML except unlike HTML, XHTML conforms to XML tagging and well-formedness rules.

Notebook: The EXI unofficial “Hello World” example XML document, intended to be a baseline example of raw XML-to-EXI conversion.

Customers and Orders: This is a common logistics XML document that contains orders for supplies, and listings of historical purchases.

Microsoft Office Suite: The Microsoft Office suite, as of the 2003 version, uses a native XML 1.0 data structure format for all of its applications (Microsoft, 2006). The new MS Office suite file format is a package of Zip-compressed XML documents that contain the contents (text, multimedia, formatting, etc.) of the underlying office file. The specific Microsoft Office case used is the notebook.xml file cut-and-paste into a Microsoft Word 2007 document.

There are three major components to Word documents:

- **Part items.** Each part item corresponds to one file in the un-zipped package. Each of those files is a document part in the package.
- **Content Type items.** Content type items describe what file types resources are stored in a document part. For example, image/jpeg denotes the JPEG images. This information enables Microsoft Office, and third-party tools, to determine the contents of any parts in the package and to process its contents accordingly.
- **Relationship items.** Relationship items specify how the collection of document parts come together to form the document; the connection between a source part and a target resource.

The folder and file-structure form a notional Microsoft Word document is depicted in Figure 52:

- **[Content_Types].xml.** Describes the content type for each part that appears in the file.
- **_rels folder.** Stores the relationship part for all parts.
- **.rels file.** Describes the relationships that begin the document structure.
- **datastore folder [optional].** Contains custom XML data parts within the document. A custom XML data part is an XML document from which you can bind nodes to content controls in the document.
- **item1.xml.** Contains some of the data that appears in the document.
- **docProps folder.** Contains the application's properties parts.
- **App.xml.** Contains application-specific properties.
- **Core.xml.** Contains common file properties.

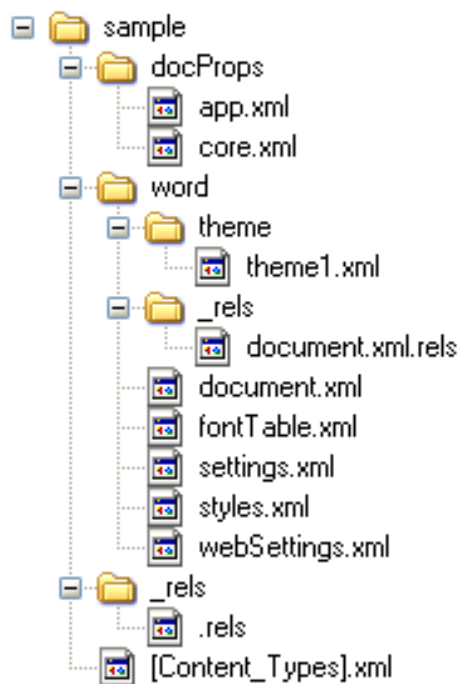


Figure 52. Microsoft Word 2007 Baseline File-Structure (From Microsoft, 2006)

3. DoD-Only Subcategory Test Cases

The following descriptions define the XML documents used for the DoD tactical messages category.

Message Text Format (MTF): MTF the XML formatting of legacy DoD digital communications. Several DoD communication-doctrine manuals, such as the ATP series and NTD series of documents describe the various message formats that the MTF language can describe.

Maritime Information Exchange Model (MIEM): The MIEM a XML-based maritime intelligence information-exchange format used to reduce the ambiguity and develop consistence between intelligence reports. The MIEM format is specifically tailored towards at-sea monitoring and maritime surveillance reports of aircraft and ship.

C. DOD-SPECIFIC EXI EXPERIMENT RESULTS

The following DoD relevant test case experiments were collected using the open source Siemens EXI engine (2009) with the default preservation options and with the compression output alignment.

For each of the three DoD subdomain categories, a text table is presented listing the effects of EXI (schemaless and schema-informed) against the commonly used DoD compression techniques: GZip and Zip. After the table, scatter plots of the same information is displayed to pictorially present the findings, both with and without the schema-informed documents, to help clarify the implications of these results. An additional set of scatter plots follow displaying the baseline effect of EXI compared directly with GZip, the current optimal compression technique for DoD.

The table structure contains 6 columns with 2 consecutive rows for each test case.

- The result percentage are rounded to the nearest integer using the mathematical rule of .5 and above round up, and otherwise rounded down.
- The result measurements are to be interpreted as the smaller the percentage of the original document, the better the compression achieved.

The columns and rows are as follows:

- File name: The first column defines the document by name as listed in the test-case definitions.
- No Operation: The second column, first row, lists the size of the original document without any compression applied. The next consecutive row of the second column lists the percentage of the original, i.e., 100% for no operation.
- GZip: The third column, first row lists the size of the input document after applying the GZip algorithm. The next consecutive row of the third column lists the percentage of the original document that the GZip algorithm produced.
- Zip: The fourth column, first row lists the size of the input document after applying the Zip algorithm. The next consecutive row of the fourth column lists the percentage of the original document that the Zip algorithm produced.
- EXI Schemaless: The fifth column, first row lists the size of the input document after applying the EXI schemaless algorithm. The next consecutive row of the fifth column lists the percentage of the original document that the EXI schemaless algorithm produced.
- EXI Schema-informed: The sixth column, first row lists the size of the input document after applying the EXI schema-informed algorithm. The next consecutive row of the sixth column lists the percentage of the original document that the EXI schema-informed algorithm produced.

1. DoD Modeling and Simulation Test Cases Results

Table 64 lists the comparison results of the DoD M&S specific category of XML test cases. Note that the MSDL, OOB, HAAR and SVG examples do not have associated schema for their XML documents. All other examples have a supporting schema.

#	FILE TYPE	ORIGINAL	GZIP	ZIP	EXI (W/O)	EXI (W)
1	MSDL	3471120 100%	262640 8%	262784 8%	58837 2%	N/A N/A
2	OOB	3420388 100%	194031 6%	194169 6%	71987 2%	N/A N/A
3	DIS Packet	1155742 100%	48887 4%	49011 4%	36160 3%	31834 3%
4	3D Map	35246 100%	2041 6%	2171 6%	1642 5%	1625 5%
5	DIS PDU	136372 100%	8545 6%	8665 6.35%	6407 5%	6351 5%
6	AVCL	10574242 100%	754744 7%	754888 7%	523560 5%	444996 4%
7	HAAR	3748256 100%	416948 11%	417116 11%	286066 8%	N/A N/A
8	Humanoid	3536 100%	567 16%	701 20%	549 16%	306 9%
9	DES Server	3034 100%	708 23%	836 28%	622 21%	451 15%
10	DES Execution	3136 100%	776 25%	920 29%	740 24%	447 14%
11	X3D	7624 100%	1551 20%	1705 22%	1898 25%	1308 17%
12	SVG	4910 100%	2238 46%	2366 48%	2178 44%	N/A N/A

Table 64. Compression Results Comparison for DoD M&S Test Case Documents

The GZip technique delivered the best compression results between the current DoD compression techniques: GZip and Zip. However, when compared to EXI schemaless, in all cases other than X3D, EXI was approximately 10% less than the next all other techniques. When compared to EXI schema-informed, in all cases EXI was less, and nearly half the size of GZip.

An interesting discovery that is not highlighted in the table is that the DIS XML format can be represented within EXI in a slightly smaller file size than the original DIS binary data format. This finding has implications for the design of future DoD protocols in that custom binary formats potentially can be replaced by EXI-encoded XML documents of the protocol data. Using XML instead of binary formats directly supports the Network-Centric data sharing strategy and increases system-to-system interoperability.

A comparison of the percentage of the original document size between Zip, GZip and EXI Schemaless and EXI Schema-informed is shown in Figure 53 and Figure 54 shows the same results with the removal of XML cases without a supporting schema. It can be seen that EXI, other than X3D under EXI schemaless, is always smaller than GZip, and in all cases EXI schema-informed rendered the smallest file.

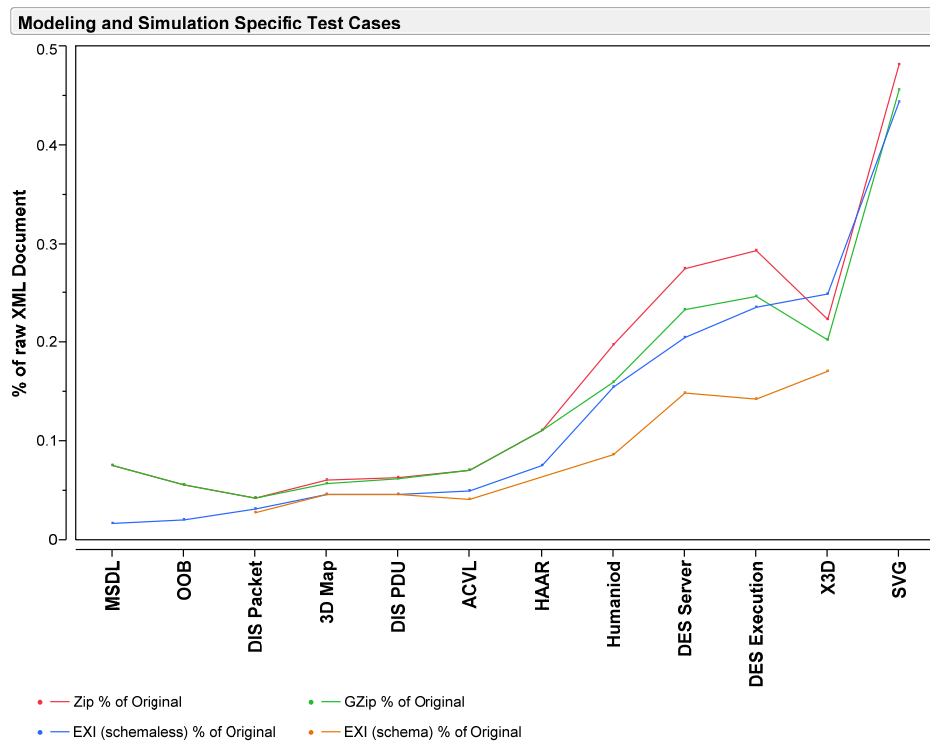


Figure 53. Comparison of EXI Encodings of the DoD M&S Test Case Documents

Remember, that the MSDL, OOB, HAAR and SVG examples do not have associated schema for their XML documents.

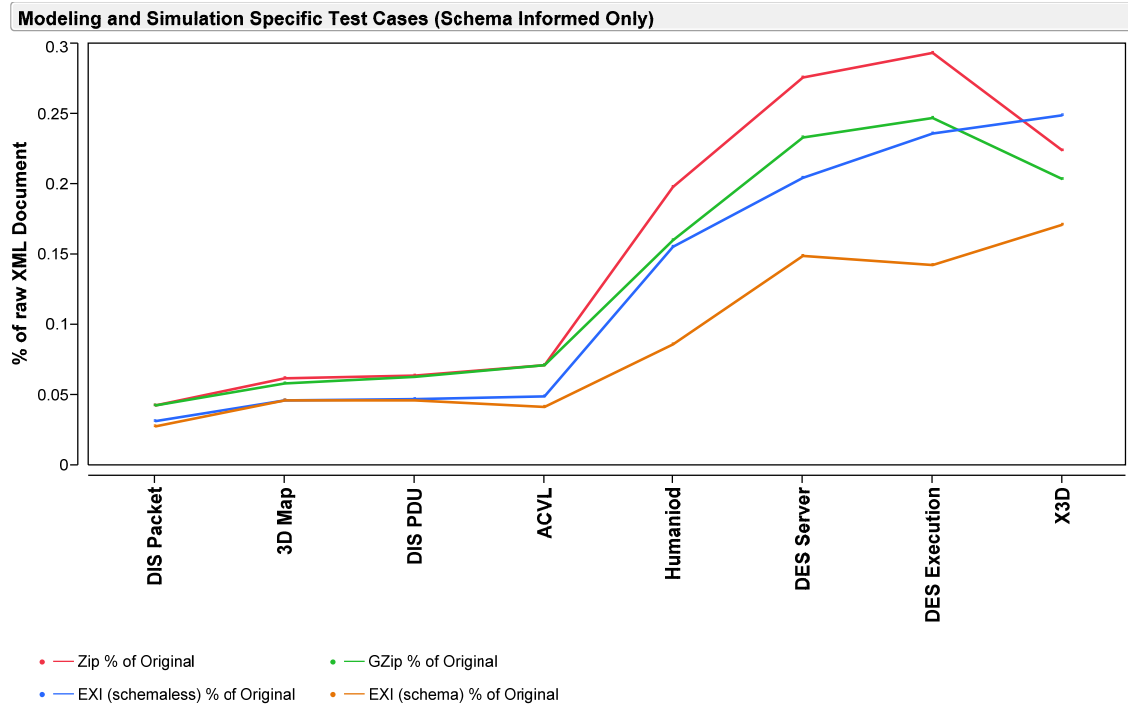


Figure 54. Comparison of EXI Schema-Informed Encodings of the DoD M&S Test Case Documents

Since GZip is the most common and optimal compression technique employed in DoD, GZip is used as the baseline comparison of EXI's true implications for DoD communications. Figure 55 shows the schemaless EXI encodings and Figure 56 shows the schema-informed XML documents as the percentage the EXI file is compared to the same GZipped file, $\frac{EXIsize}{GZIPsize}$. On average, an EXI schemaless file is 78% ($\pm 12\%$ at .95) of the size of a GZIP file, and the EXI schema-informed file is 62% (± 6 at .95).

Based on these results, EXI has the potential of delivering a bandwidth increase of 61% for schema-informed, and 28% for schemaless based on the rate of change equation

$$\left(\frac{1}{EXI\%GZip} - 1 \right) 100.$$

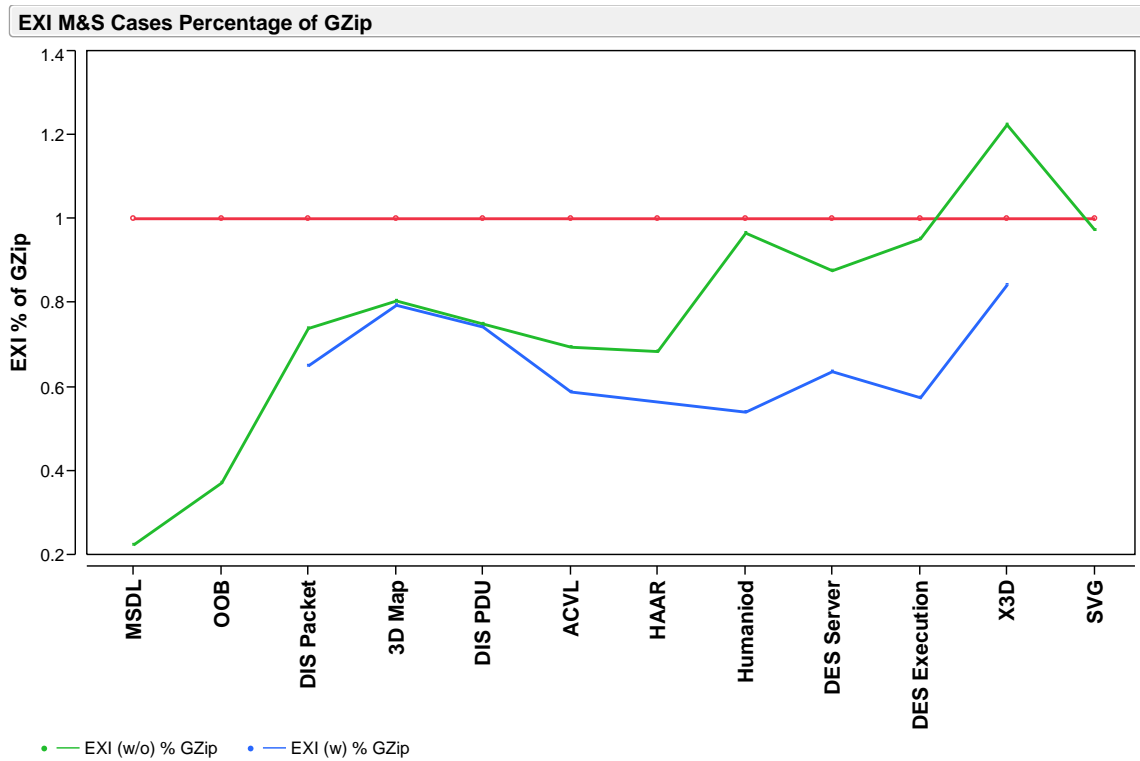


Figure 55. Comparison of EXI Encodings Baselined to GZip for the DoD M&S Test Case Documents

Note that the MSDL, OOB, HAAR and SVG examples do not have associated schema for their XML documents.

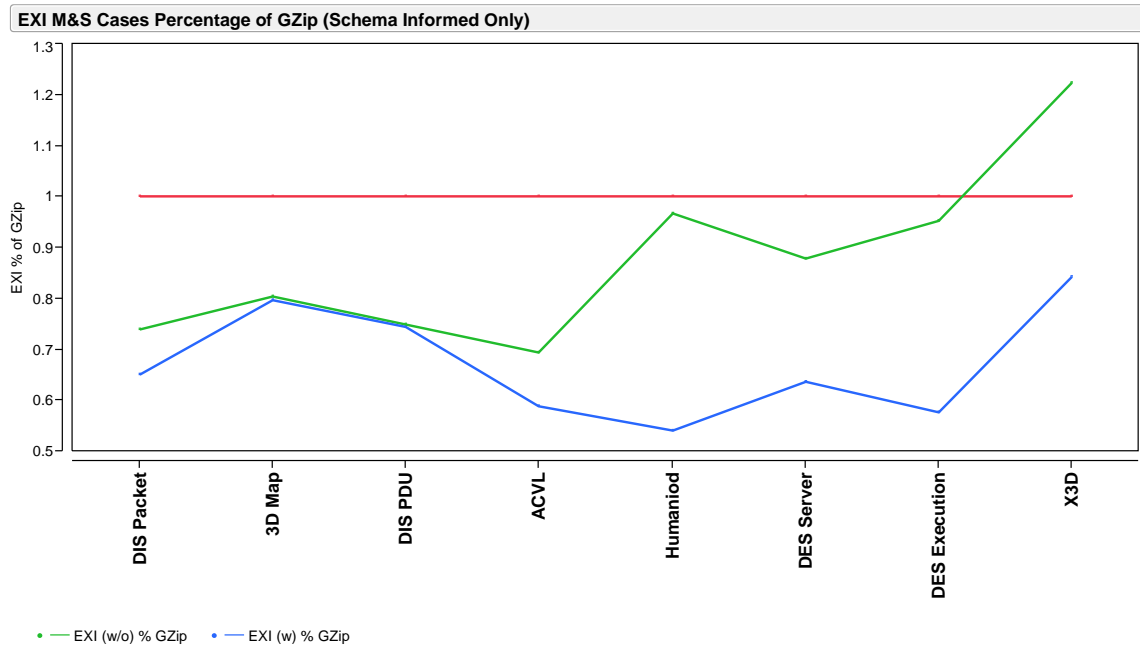


Figure 56. Comparison of EXI Schema-Informed Encodings Baselined to GZip for the DoD M&S Test Case Documents

2. DoD General Test Cases Results

Repeating the same experiment as conducted for the DoD M&S category results, Table 65 is constructed with the same design format for the general DoD category of test cases. Note that the Word 2007 and the logistics examples do not have an associated schema for their XML documents.

#	FILE TYPE	ORIGINAL	GZIP	ZIP	EXI (W/O)	EXI (W)
1	DOCX:document.xml	13678 100%	908 7%	1028 8%	682 5%	N/A N/A
2	DOCX:styles.xml	14746 100%	1466 10%	1582 11%	1334 9%	N/A N/A
3	WEB:xhtmlSpec.xml	293493 100%	34189 12%	34311 12%	33623 11%	33373 11%
4	WEB:msn.xml	99298 100%	16656 17%	16766 17%	14879 15%	14307 14%
5	LOGISTIC:customers.xml	2257 100%	678 30%	800 35%	479 21%	N/A N/A
6	DOCX:theme1.xml	6992 100%	1492 21%	1608 23%	1509 21%	N/A N/A
7	WEB:w3cWebpage.xml	68770 100.00%	13915 20%	14039 20%	14996 22%	14474 21%
8	LOGISTIC:order.xml	1547 100%	554 36%	668 43%	383 25%	N/A N/A
9	DOCX:fontTable.xml	1295 100%	410 32%	532 41%	326 25%	N/A N/A
10	DOCX:document.xml.rels	817 100%	255 31%	385 47%	213 26%	N/A N/A
11	DOCX:[Content_Types].xml	1312 100%	359 27%	489 37%	343 26%	N/A N/A
12	DOCX:core.xml	702 100%	356 51%	468 67%	238 34%	N/A N/A
13	DOCX:.rels	590 100%	251 43%	357 61%	202 34%	N/A N/A
14	DOCX:webSettings.xml	260 100%	198 76%	324 125%	93 36%	N/A N/A
15	DOCX:settings.xml	1755 100%	749 43%	869 50%	733 42%	N/A N/A
16	HELLOWORLD:notebook.xml	321 100%	196 61%	316 98%	135 42%	68 21%
17	DOCX:app.xml	987 100%	482 49%	592 60%	419 42%	N/A N/A

Table 65. Compression Results Comparison for General Use DoD Test Case Documents

Again, like the DoD M&S results, EXI delivered the best overall compression. However, an unique occurrence happened within this case set, the WEB:w3cWebpage.xml document was compressed better with GZip than EXI schema-informed by 0.8 of a percent. This is a minor difference, but of all the tested documents from the three test categories, it is the only document where EXI schema-informed was outperformed.

In the more general case observations, only the WEB:w3cWebpage.xml and the DOCS:theme1.xml test case files did the EXI schemaless technique resulted in a larger compressed file than GZip, though just slightly. The Zip technique delivered on two occasions with a resulting file greater than the original document, DOCX:webSettings.xml and HELLOWORLD:notebook.xml. General results conclusion is that with or without a schema, EXI delivers noticeable file size savings averages compared to GZip, and far exceeded the Zip results.

A side-by-side comparison of the percentage of the original file size between Zip, GZip and EXI is shown in Figure 57 and Figure 58, with the schema-informed cases only. Other than other than the DOCX:theme1.xml and WWW:w3cWebpage.xml , both EXI schemaless and EXI schema-informed always deliver a result file less than or equal to all other techniques result.

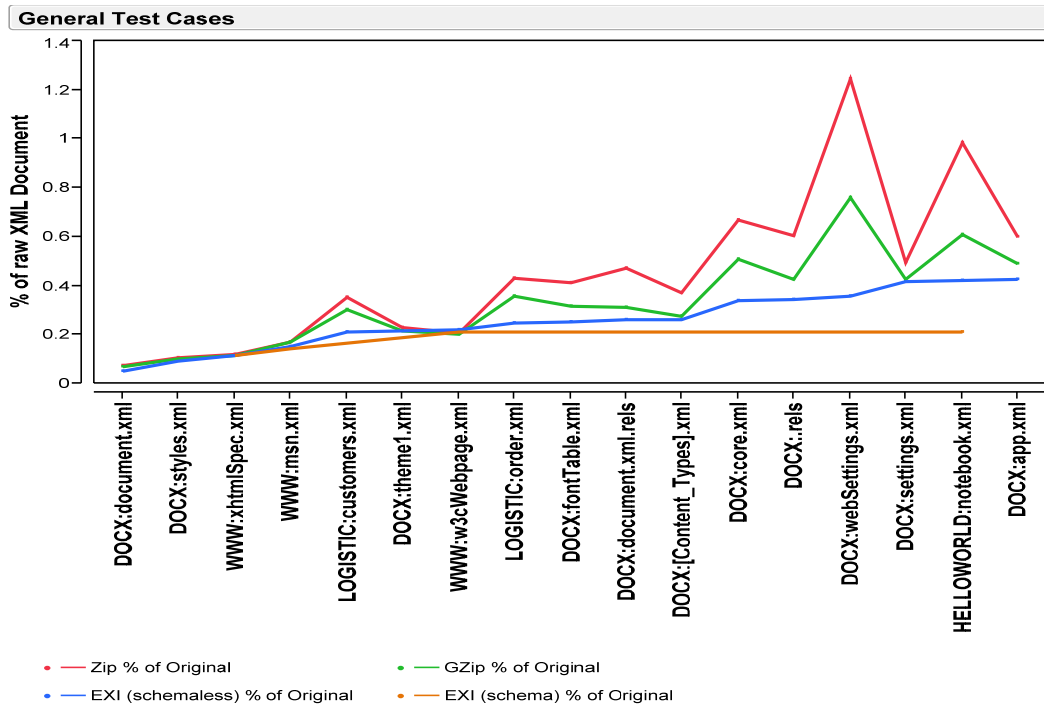


Figure 57. Comparison of EXI Encodings on the DoD General Test Case Documents

Note that the Word 2007 (DOCX) and the logistics examples do not have an associated schema for their XML documents.

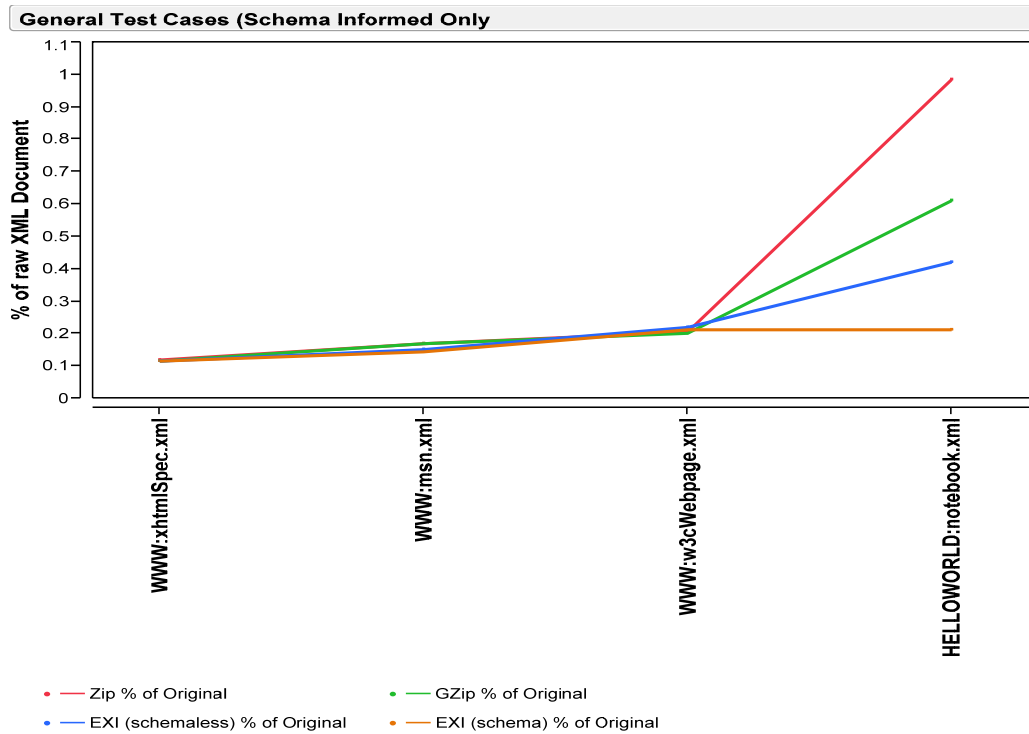


Figure 58. Comparison of EXI Schema-Informed Encodings of the DoD General Test Case Documents

Continuing with the fact that GZip is the most optimal compression technique employed in DoD, it is used as the baseline comparison of EXI's overall effectiveness in delivering a compact XML file for DoD. Figure 59 and Figure 60 schema-informed cases only, shows the percentage of EXI to that of the same GZipped file. The overall average percentage of EXI/GZip for schemaless EXI is 82% (± 6 at .95), and for schema-informed 80% (± 25 at .95). Because the number of schema-informed samples for the general cases was so small, any reasonable measure of significance is impossible and is noted in the wide confidence interval. However, the result averages are consistent with experiments with larger sample sizes.

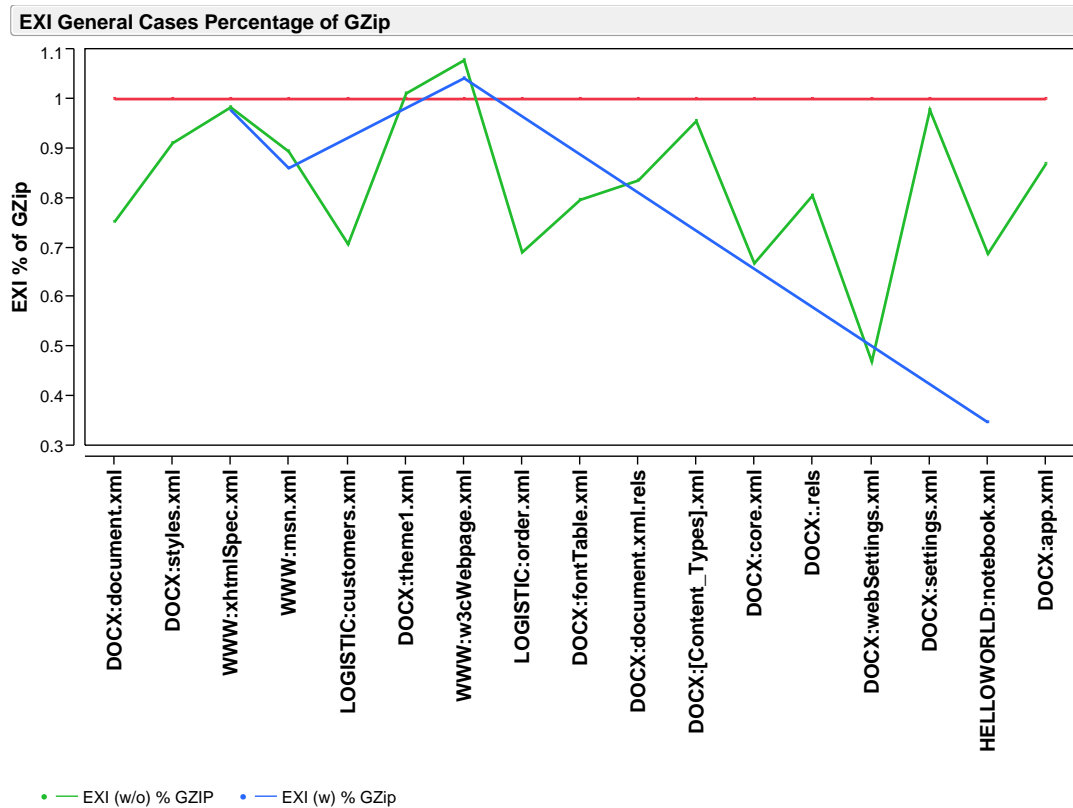


Figure 59. Comparison of EXI Encodings Baselined on GZip for the DoD General Test Case Documents

Note that the Word 2007 (DOCX) and the logistics examples do not have an associated schema for their XML documents.

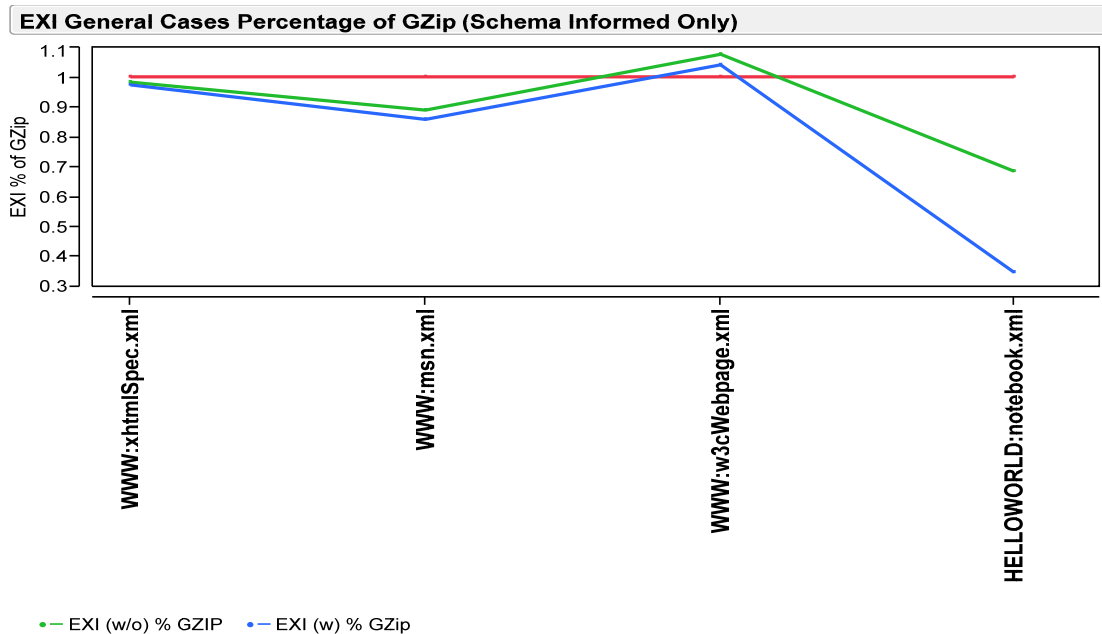


Figure 60. Scatter Plot Comparison of EXI Schema-Informed Encodings Baselined on GZip for the DoD General Test Case Documents

An interesting note about the Word 2007 file is that if the original 10.4KB .docx file is directly compressed with Zip, it results in an 8KB file, but the sum total of the EXI files that make up the .docx file is only 5.61Kb, or 70% of the compressed.docx file. This is another positive indicator of the effectiveness of the EXI technique to the entire XML family of languages as well as structured XML packages. Additionally, this highlights the general bandwidth savings potential of the EXI technique in support of the GIG and Network-Centric XML data sharing strategy. This is especially important as more and more XML documents as well as XML packages are incorporated into the GIG.

3. DoD-Only Test Cases Results

Repeating the same experiments as conducted for previous categories, the DoD-specific category comparison compression Table 66 is constructed with consistent table formatting. Note that all cases other than iso_3166-1_list_en.xml, ExemplarCommsPlan.xml, confrencePlanningMSG.xml, farewellMSG.xml and visitRequest.xml have a supporting schema.

#	FILE TYPE	ORIGINAL	GZIP	ZIP	EXI (W/O)	EXI (W)
1	iso_3166-1_list_en.xml	45006 100%	2970 7%	3110 7%	2152 5%	N/A N/A
2	FacilityTestCase.xml	5984 100%	590 10%	726 12%	390 7%	170 3%
3	Vignettes.xml	31942 100%	6413 20%	6535 20%	3585 11%	2272 7%
4	ShippingOrganizationList.xml	6046 100%	1095 18%	1247 21%	860 14%	399 7%
5	visitRequest.xml	6258 100%	1695 27%	1823 29%	1084 17%	N/A N/A
6	VesselSummary(ELONA).xml	37732 100%	7428 20%	7592 20%	6748 18%	5219 14%
7	ExemplarCommsPlan.xml	27793 100%	5650 20%	5788 21%	5098 18%	N/A N/A
8	Snippet3-21.xml	24959 100%	5030 20%	5156 21%	4679 19%	3121 13%
9	VesselCaseFile.xml	25530 100%	5245 21%	5377 21%	4893 19%	3371 13%
10	TestCase-7.xml	3313 100%	1069 32%	1193 36%	654 20%	354 11%
11	confrencePlanningMSG.xml	19725 100%	4960 25%	5104 26%	4045 21%	N/A N/A
12	farewellMSG.xml	3349 100%	1336 40%	1462 44%	795 24%	N/A N/A
13	PortsWithBerthsEx1.xml	1785 100%	626 35%	766 43%	430 24%	221 12%
14	CurrencyTest.xml	525 100%	300 57%	428 82%	153 29%	92 18%
15	TestCase-0.xml	424 100%	305 72%	429 101%	128 30%	74 17%
16	PosEx1.xml	1506 100%	626 42%	742 49%	582 39%	260 17%

Table 66. Compression Results Comparison for DoD-Specific Test Case Documents

A comparison of the percentage of the original file size between Zip, GZip, EXI schemaless and EXI schema-informed for the DoD-Specific documents is shown in Figure 61, and Figure 62 for the schema-informed cases only. In all cases, EXI, both schemaless and schema-informed, delivered the smallest files.

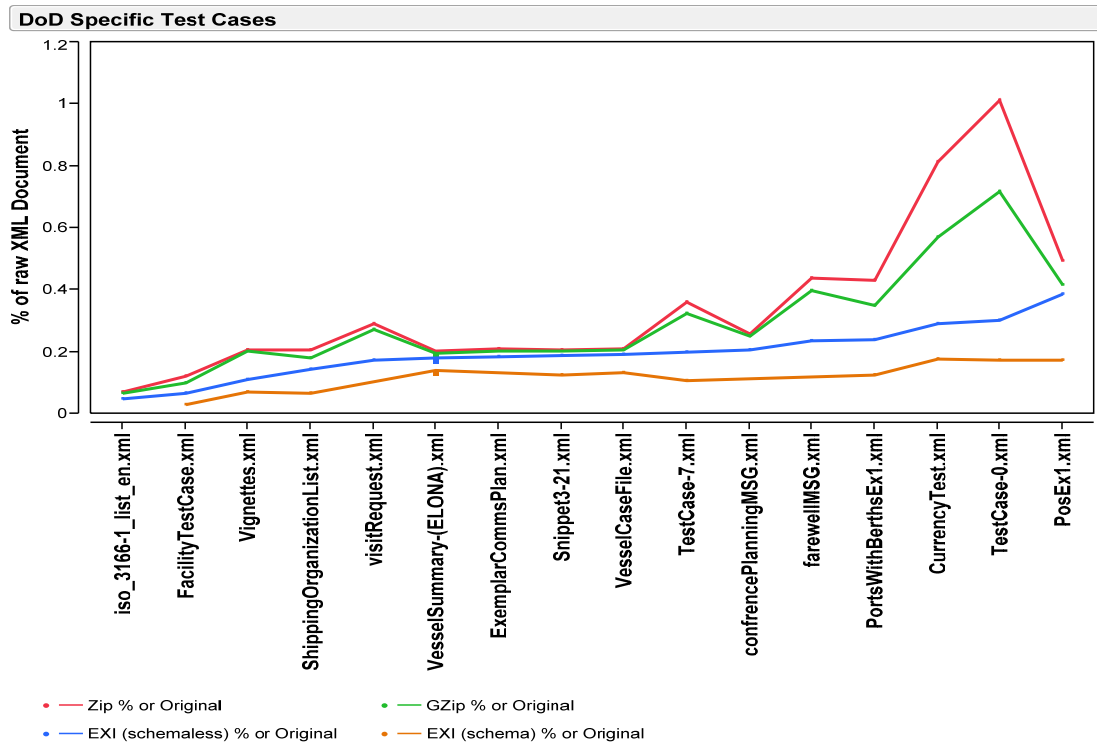


Figure 61. Comparison of EXI Encodings of the DoD-Specific Test Case Documents

Note that all cases other than iso_3166-1_list_en.xml, ExemplarCommsPlan.xml, conferencePlanningMSG.xml, farewellMSG.xml and visitRequest.xml have a supporting schema.

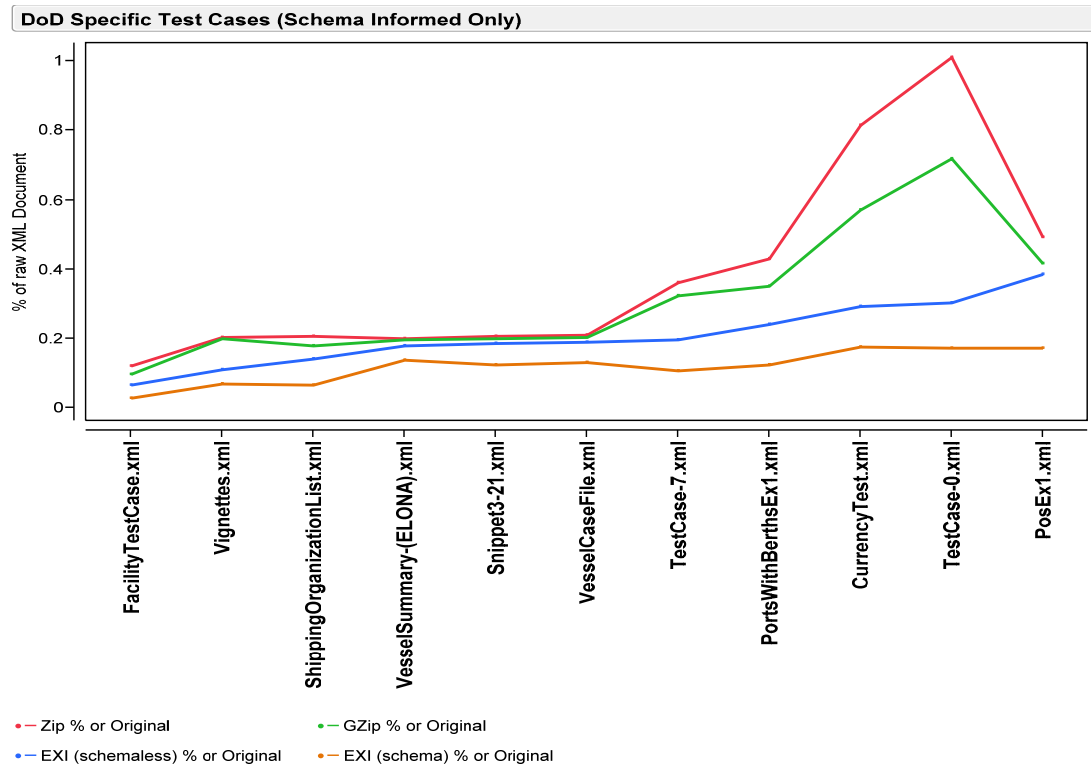


Figure 62. Comparison of EXI Schema-Informed Encodings of the DoD-Specific Test Case Documents

Figure 63 and Figure 64 for the schema-informed cases only, shows the EXI results of the DoD-Specific cases as a percentage of GZip. On average the DoD only cases of schemaless EXI deliver a file of 72% (± 7 at .95) of GZip, and the schema-informed EXI file is 42% (± 8 at .95) of GZip. It is important to notice the narrow confidence interval. This indicates the likelihood of consistent results for DoD-Specific documents. As was highlighted in the framework testing of candidate compressed binary XML formats the military consistently had a supporting schema for its documents, which a schema always delivers a and consistent EXI performance.

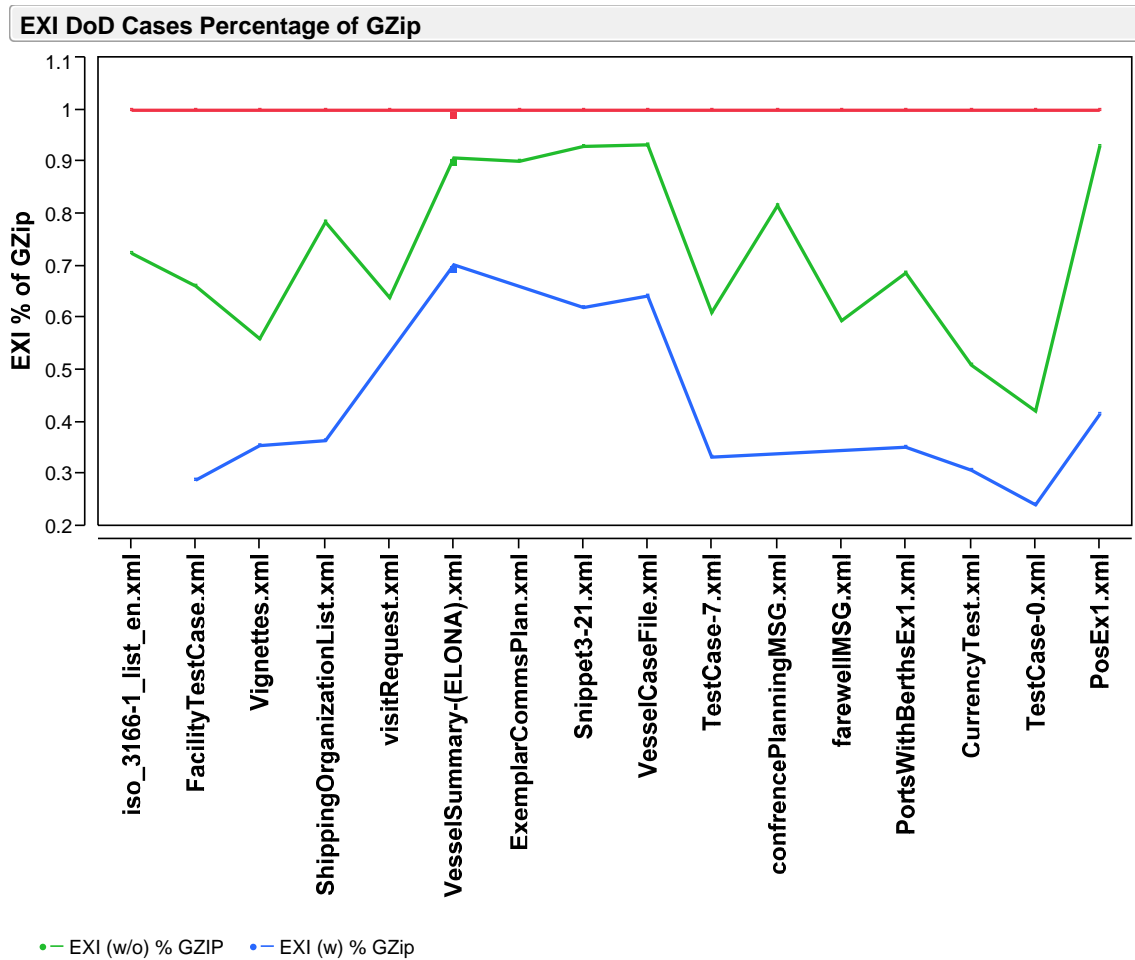


Figure 63. Comparison of EXI Encodings Baselined on GZip for the DoD-specific Test Case Documents

Note that all cases other than iso_3166-1_list_en.xml, ExemplarCommsPlan.xml, conferencePlanningMSG.xml, farewellMSG.xml and visitRequest.xml have a supporting schema.

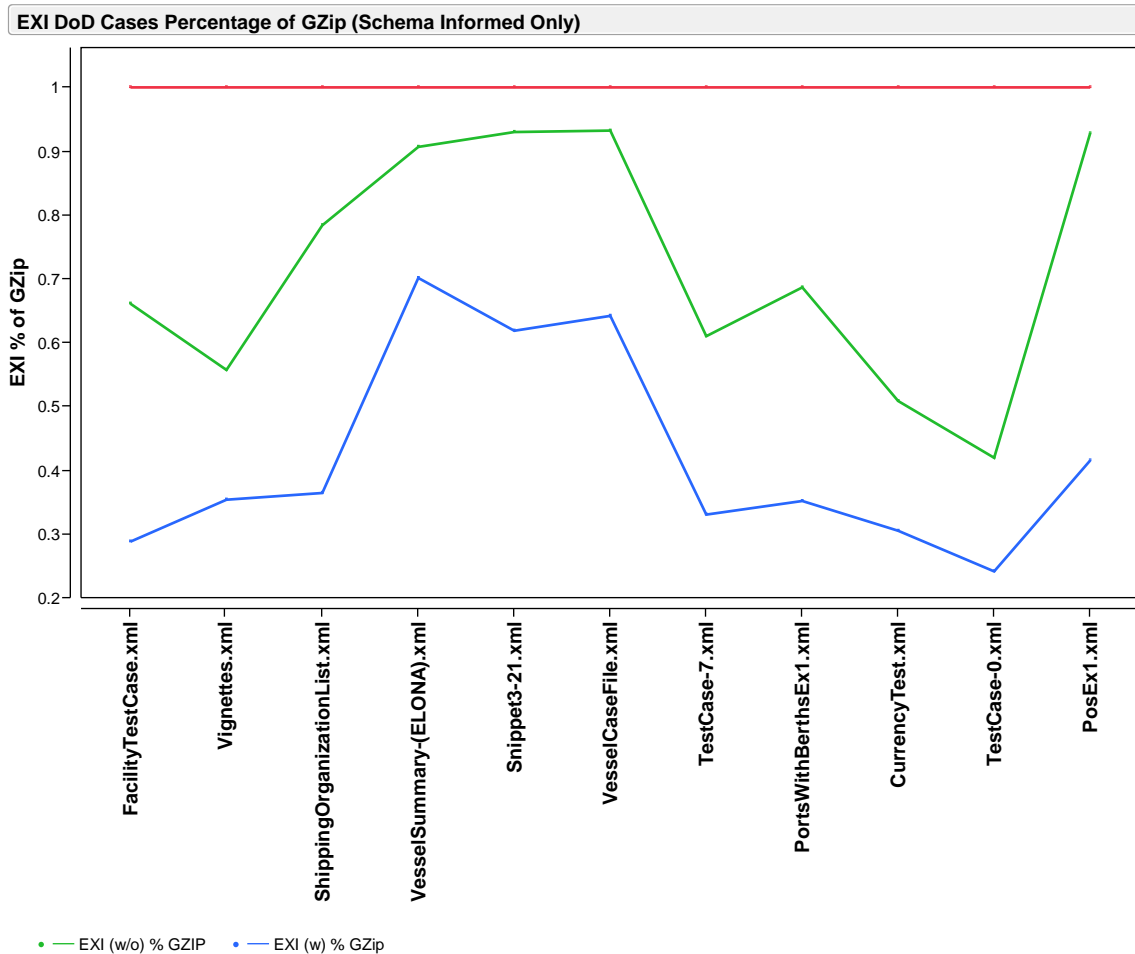


Figure 64. Comparison of EXI Schema-informed Encodings Baselined on GZip for the DoD-specific Test Case Documents

The significance of these EXI findings for the DoD-specific test cases are the increase in bandwidth potential EXI can deliver within the low-bandwidth environments that DoD operates. For schemaless XML documents the resulting file average is 72% of the baseline transmitted GZip file, which is approximately a 50% increase in bandwidth potential without altering the network architecture such as new satellite connections and additional hull mounted satellite antennas. The schema-informed cases resulting file size average is 42%, which is a 118% increase in bandwidth potential of the same file under GZip. Simply put, two EXI files can be transmitted in less time than a single GZip file.

4. DoD EXI Test Cases Summary

EXI has the potential to increase valuable bandwidth to remote stations that operate under low to extremely low-bandwidth conditions.

A notable mention that is repeated in many DoD XML usage studies is that DoD's uses of schemas as well as XML documents that abide by the schema, is far better than the IT industry in general. This means DoD is primed to receive maximum bandwidth gains potential from EXI.

Given DoD's strong schema awareness, the compactness of DoD specific files compressed with schema-informed EXI is the best measurement of likely DoD experience. EXI therefore, has an $E[X] \geq 1$ or over 100% increase in bandwidth potential, which is a doubling bandwidth benefits for the DoD.

5. W3C Corpus of Results

Additional results can be obtained from the EXI Working Group's *Efficient XML Interchange Evaluation* document (W3C, 2008) and *Efficient XML Interchange Measurements Note* (W3C, 2007). The results shown here agree with the findings of the EXI Working Group, which found that EXI is consistently (and often remarkably more) compact than XML documents encoded with GZip and other compression formats.

D. RECOMMENDED EXI CONFIGURATION

The following recommended EXI parameter settings are specifically focused towards optimizing the DoD domain, but hold merit in general to all XML domains. The focus is maximum compactness given DoD XML domain is more concerned with bandwidth improvements than efficiency. Additionally, as Mores Law continues and hardware improves, bandwidth will likely be the only hurdle XML will have to continue to overcome; EXI resolves the bandwidth hurdle.

1. Use of a Schema Whenever Possible

Always use schema-informed EXI encodings whenever a schema exists. For XML documents cases that do not have a schema, a schema ought to be created.

A schema enables the most compact representation of XML using the EXI compression algorithm. Due to encoding limitations during the learning process for a schemaless XML document, the level of compactness of schemaless EXI cannot surpass or equal that of a schema-informed document. However, the ability to encode XML without a schema is a W3C requirement because many XML documents do not have a schema, or if they do, often do not fully comply with its schema. While schema-informed encoding does deliver the best results, schemaless encoding usually delivers GZip or better levels of compactness. Further, there is significant benefits to ensure that created XML documents are valid prior to transfer in order to minimize wasted processing efforts and avoid Garbage In Garbage Out (GIGO) syndrome.

a. N-bit Minimization

An XML schema ensures the smallest N for N-bit event codes, grammar indexes and string table indexes.

Without a schema, the number of events in a grammar is unknown until the initial EXI parsing process completes because the encoding algorithm is always learning until an EndDocument event is fired. Additionally, the specific types of events that can be encoded into the grammar remain unknown until the EndDocument event is fired.

The EXI encoding algorithm can exploit XML schema information to create grammars with the smallest compact identifiers. A schema in essence creates the first occurrence of every element, attribute, and datatype that can be encountered in an XML document before the document is processed by the EXI processor. This determines optimized encodings immediately without the need for learning by pre coding the grammar size and event types for all grammars, string tables namespaces and local-name entries.

b. Datatype Binding

An XML schema defines the datatypes of the XML document. EXI uses this information to write binary representations of numeric types that are smaller than text representation, and often smaller than traditional IEEE numeric formats.

Without a schema, EXI encoding is limited to the string datatype only. Numeric, date, and other datatypes can only be represented as strings, which is not efficient. Retaining numeric values in a more compact binary format enables both file size savings and reduced processor complexity by eliminating the need for complex string-to-number parsing.

EXI's variable-length numeric-datatype binary encoding can represent numeric values in fewer bits than fixed-length numeric value formats such as Integer IEEE, floating-point or double precision formats. For example, integers with a small value can be represented in EXI with a single byte, and larger integer values are represented in multiple bytes, but only just enough bytes to represent the number. An IEEE binary integer on the other hand is always 4 bytes (32 bits) regardless of the value it represents; the value one takes up as much space as does one-million or one-billion, while EXI uses just enough bytes to represent the number.

Ultimately, EXI's binary datatypes are more compact than strings, IEEE numeric types or other fixed-byte-size datatype formats.

2. No Preservation Options Settings

Without express need, pruneable events need not be preserved. In other words, default preservation settings of false. The goal of EXI is to generate the most compact and efficient XML representation possible while retaining the data of the XML document. Quite often, these pruneable events are not essential to the meaning of a document and can be discarded while at the same time keeping the document's original functional state. Details of this conclusion follow.

a. Comments

Comments typically add no intrinsic value to XML documents in transmission or machine processing. Comments are usually only for the human user or for adaption a document for reuse as a new document. Often an EXI implementation operates in a Web services or other data-transfer arena, and so the XML is not intended to be directly read by humans; the XML is feeding another system that does not use comments. Retaining comments bloats EXI.

Unless the EXI transfer is explicitly intended for human reading or subsequent editing, the preservation of comments need not be used.

b. Namespaces

Specific namespace prefix naming has no intrinsic value to the XML document other than a unique prefix must be associated to each URI. Given this, not retaining namespace prefixes at EXI encoding, and then automatically generating a unique prefix at decoding (ns1, ns2,...) eliminates the added event bits, characters, and namespace flag values from being written to the EXI stream. Not preserving namespaces does not usually result in a noticeable savings given prefixes are generally small, only 2 to 4 characters, though any file size savings that can be realized are useful if utilized. Mnemonic namespace values can help humans reading or edition a document.

Without an explicit need to preserve a particular prefix naming convention, namespace events need not be preserved within an EXI stream in favor of processor auto generated prefixes at decode.

c. Entity and DTD

First and foremost, XML documents that rely on Entities and DTDs need to be updated to a schema-based validation structure. However, if the XML to be encoded with EXI uses Entities and or DTDs, they need not be preserved within the EXI stream.

Within EXI, Entity and DTD events are written to the EXI stream in native byte-aligned text, which is basically a cut-and-paste of their declaration directly into the EXI stream from the XML document. This significantly bloats the size of an EXI stream. Most importantly, the flexibility of type encodings of Entities and DTDs are limited to just defining string sequences, which these sequences can be learned by the EXI grammar algorithm regardless of the DTD or Entity reference.

Without a unique necessity, Entities and DTDs need not be preserved within an EXI stream.

d. Processing Instructions (PI)

Processing instructions present a possible exception to the rule about not preserving pruneable options. Some XML processing instructions are essential to proper presentation of XML data such as the xml-stylesheet instruction (`<?xml-stylesheet type="text/xsl" href="xml_HTML.xslt" ?>`) which enables XML formatting for presentation within an HTML browser. However, most processing instructions are system specific and can be inferred when processing the XML documents. For example, if an XML document is going to be processed by an XSLT Stylesheet for formatting to an HTML Web browser viewable format, this is generally known before processing, and as such can be implied and not included in the EXI stream. The case where this may not apply is for ad-hoc XML documents that are not going to predefined destinations or services, in which cases, the processing instructions would not be able to be extrapolated requiring the preservation of processing instruction events to ensure proper presentation or processing of the XML document. Many Naming and Design Rules (NDR) preclude or prohibit the use of PI because they are system-specific and not general.

For routine XML system-to-system traffic, processing instruction (PI) events need not be retained with the intent of the receiving station being able to imply the instruction needs out of the routine nature of the transaction. For non-routine XML exchanges, the processing instructions may have to be retained depending upon the severity of disregarding the instruction.

3. Use Post-Processing Compression

The added step of EXI compression substantially adds to the overall net compression savings of the EXI technique and is usually best used outside of fragmented streaming data transactions.

In no case did the execution of EXI compression deliver a file size larger than without compression. Because the compression algorithm used within EXI is based on the rather simple and well understood INFLATE/DEFLATE algorithm (RFC 1950 and 1951) the added complexity of EXI compression is trivial, and should always be used unless explicitly advised otherwise.

The focus of EXI is to deliver the most compact and efficient XML binary XML format, and when the base EXI streaming technique is paired with EXI compression, no other compression algorithm can come close to matching it.

It should be highlighted that even the XML cases where encryption and digital signature are needed, EXI compression remains valid and effective. This holds base on the fact EXI works on the XML Information set, well-formed XML, and not the raw data. Encryption and digital signature encapsulate XML data into another XML document, maintaining a well-formed and valid XML document. This equates anywhere within the XML Infoset stack, EXI and EXI compression remain valid techniques.

The only possible exception to the ubiquitous use of EXI compression is when dealing with streaming data. Streaming XML data is generally small fragments of XML at high update rates. The small size of the streaming fragments will not likely have many redundant characters within the fragment. The compression of these fragments will not likely lead to significant gains in compression to justify the added complexity. Additionally, depending on the update rate of streaming data, the fragments may be too rapid for the added complexity of EXI compression to keep pace with the inflow of data. Nevertheless type compression for numeric datatypes may still compress some fragments satisfactorily.

4. Experiment of the Recommended EXI Configuration Settings

To demonstrate the recommended parameter settings, Table 67 lists (in decreasing resulting file size order) several EXI encoding configurations of the HelloWorld.x3d X3D document which itself is listed in Table 68. HelloWorld.x3d is encoded in different ways in terms of schema (2 setting states: schema-informed and schemaless), Alignment (3 setting states: bit, byte and compression), and Preservation (2 setting states: all options pruned and all options preserved), for a full factorial experiment of $2 \times 3 \times 2 = 12$ comparable cases of the same file. The percentage of original results are rounded to the nearest whole integer, and the smaller the percentage the better the results.

EXI options Used			Resulting Size	% Original
Schema	Alignment	Preservation		
Original file	Original File	Original File	2025	100%
No	Byte	All	1758	87%
No	Byte	None	1692	84%
No	Bit	All	1552	77%
No	Bit	None	1485	73%
Yes	Byte	All	1101	54%
Yes	Byte	None	1033	51%
Yes	Bit	All	1001	49%
No	Compress	All	981	48%
No	Compress	None	930	46%
Yes	Bit	None	926	46%
Yes	Compress	All	607	30%
Yes	Compress	None	542	27%

Table 67. Rowhead Parameter-Dependent Results of Full-Factorial Compression Experiment

a. Preservation

In all cases, regardless of the schema setting or the alignment setting, the no preservations encodings or all options pruned, the default EXI configuration setting delivered a smaller resulting file than when all options were preserved. This result is intuitively obvious given the optional data was not retained in the EXI encoding.

b. Alignment

As might be expected, the alignment results went in decreasing file size order of byte, bit, then compressed. This again, like the preservation of options, is intuitively obvious in that a byte is larger than a bit, and compressed bits are less than sequential bits with padding.

c. XML Schema

Again, as expected, a schema-informed document delivered a smaller resulting file than a schemaless document. One result that was not initially expected was the schemaless with compressed aligned output delivered comparable results to the bit-aligned schema-informed output. This simply highlights the fact that schema-informed type compression always ought to be used.

d. Conclusion of Recommended EXI Configuration Experiment

These results confirm what was obvious at the start: a compressed, schema-informed file that does not preserve optional information will deliver a smaller file, larger compression ratio, than any other configuration for the EXI technique. Given these intuitive results, and with the focus of generating the smallest file possible to maximize limited DoD bandwidth, without an explicit to the contrary need, all EXI files ought to be encoded with a schema when possible, using the compressed alignment, and without preserving optional events.

An additional comment of the results from Table 67 is that even a file without a schema under a compressed alignment delivers nearly equal results to that of a schema-informed bit aligned output. This finding indicates that the alignment of compression is the most significant setting; compression alignment must be used to achieve optimal bandwidth maximization.

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- A sample case of the hello world.x3d example to demonstrate
the recommended EXI parameter settings -->
<?xml-stylesheet type="text/xsl" href="xml_HTML.xslt" ?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN"
"http://www.web3d.org/specifications/x3d-3.1.dtd">
  <X3D profile='Immersive' version='3.1'
xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifications/x3d-
3.1.xsd'>
    <head>
      <meta content='HelloWorld.x3d' name='title'/>
      <meta content='Simple X3D example' name='description'/>
      <meta content='30 October 2000' name='created'/>
      <meta content='20 December 2007' name='modified'/>
      <meta content='Don Brutzman' name='creator'/>
      <meta
content='http://www.web3d.org/x3d/content/examples/HelloWorld.x3d'
name='identifier'/>
      <meta content='X3D-Edit 3.2, https://savage.nps.edu/X3D-Edit'
name='generator'/>
      <meta content='license.html' name='license'/>
    </head>
    <Scene>
      <!-- Example scene to illustrate X3D tags and attributes. -->
      <Group>
        <Viewpoint centerOfRotation='0 -1 0' description='Hello world!'
position='0 -1 7'/>
        <Transform rotation='0 1 0 3'>
          <Shape>
            <Sphere/>
            <Appearance>
              <Material diffuseColor='0 0.5 1'/>
              <ImageTexture url='earth-topo.png' "earth-topo.jpg" "earth-
topo-small.gif" "http://www.web3d.org/x3d/content/examples/Basic/earth-topo.png"
"http://www.web3d.org/x3d/content/examples/Basic/earth-topo.jpg"
"http://www.web3d.org/x3d/content/examples/Basic/earth-topo-small.gif"'/>
            </Appearance>
          </Shape>
        </Transform>
        <Transform translation='0 -2 0'>
          <Shape>
            <Text solid='false' string='"Hello" "world!"'>
              <FontStyle justify='MIDDLE' "MIDDLE"'/>
            </Text>
            <Appearance>
              <Material diffuseColor='0.1 0.5 1'/>
            </Appearance>
          </Shape>
        </Transform>
      </Group>
    </Scene>
  </X3D>

```

Table 68. Recommended EXI Configuration Experiment Test Document
HelloWorld.x3d (From Brutzman, 2007)

E. TEST-CORPUS FOR STATISTICAL COLLECTIONS

To test the significance between techniques, and to generate predictive EXI models, a test-corpus of XML documents is compiled from the gamut of XML languages. The test-corpus consist of a total of 773 documents, based on the W3C working group test-corpus hosted by NPS (Table 13) and extended with the following Use Groups and Test Cases highlighted in bold font within Table 69. The test corpus itself has been culled from several thousand XML documents in order to eliminate documents with duplicate characteristics. For each extension, a brief description follows the table.

Category	Description	Cases included
Scientific information	This covers data that is largely numeric in nature, used in scientific applications.	GAML, HepRep MAGE-ML, XAL, HARR
Financial information	This use group includes cases in which the information is largely structured around typical financial exchanges: invoices, derivatives, etc.	FixML, FpML, Invoice, XBRL
Electronic documents	These are documents that are intended for human consumption, and can capture text structure, style, and graphics.	OpenOffice, SVGTiny, Factbook, Docbook, DOCX, MARCXML, XPDF
Web services	This use group consists of documents related to Web services, both messages and other types of documents.	Google, WSDL, RosettaNet, RSS, SOAP, UBL, XHTML, WS-Addressing
Military information	These documents are encountered in military use cases.	AVCL, ASMTF, JTLM, MIEM, TacMsg
Broadcast metadata	The type of information in this use group captures information typically used in broadcast scenarios to provide metadata about programs and services (e.g., title, synopsis, start time, duration, etc.).	CBMS, BCAST
Data storage	This use group covers data-oriented XML documents of the kind that appear when XML is used to store the type of information that is often found in RDBMS.	DataStore and Periodic
Sensor information	Documents in this use group are information potentially provided by a variety of sensors.	Seismic, epicsArchiver, LocationSightings
Scripting tools	This group includes cases of scripting and make like files.	Ant
Semantic Web	This group includes cases of modern descriptive meta tags to XML documents.	DAML, RDF
Modeling and Simulation	This group includes cases of simulation definitions, parameter and state configuration, and visualization representation	Delta3D, DIS, MSDL, OOB, VISKIT, X3D
Medical	This group includes cases of XML usage by the medical communities	MeSH, NLM
General Cases	This group includes general XML files	Notebook, points

Table 69. Extensions of the W3C Test-Corpus (After W3C, 2008)

- Ant–Ant is a Java-based build tool from the Apache Software Foundation, similar to make functions but in a XML format.
- BCAST–It is a XML-based broadcast media format used by mobile and handheld devices to transmit and display media content (TV, weather,...).
- DAML–DARPA Agent Markup Language is a XML language for describing semantic Web ontologies.
- DIS–Is a XML binding of the IEEE Distributed Interactive Simulation protocol used to distribute entity state and actions between simulation participants.
- Docbook–Is a XML format used to publishing Web-based documents that have the look and feel of physical text-based books.
- Delta3D–Is an open source game engine that uses XML as the scene and entity state definition format.
- DOCX–Microsoft Office Suite XML format, specifically Word in this case.
- HAAR–OpenCV artificial intelligence vision package uses XML to configure its real-time facial detection algorithm and as its data state persistence storage format.
- MARCXML–Is a XML language used by the US library of Congress to perform literal conversion of MARC to XML. The MARC format is the standard format for representing and communicating bibliographic and related information in machine-readable form.
- MeSH–Medical Subject Headings is the U.S. National Library of Medicine's controlled vocabulary thesaurus in XML format.
- MIEM–Maritime Information Exchange Model is an XML-based intelligence data exchange format.

- MSDDL–Military Scenario Definition Language provides a standard mechanism for loading Military Scenarios independent of the application generating or using the scenario.
- NLM–A set of standard XML vocabularies from the National Library of Medicine for medical data records and data exchange.
- OOB–Is a simulation Order of Battle definition in XML format that defines the starting state of a simulation objects.
- RDF–Resource Description Framework is a general-purpose language for representing information in the Web.
- RSS–Really Simple Syndication is a XML format for syndicating news and the content of news-like sites.
- RosettaNet–The RosettaNet standard is based on XML that defines message guidelines, business processes interface and implementation frameworks for interactions between companies.
- SOAP–Simple Object Access Protocol is a protocol specification for exchanging structured information in the implementation of Web Services within computer networks.
- TacMsg–Tactical military messages in XML format.
- Viskit–A visual Discrete Event Simulation (DES) Engine that uses XML as the simulation state and parameter configurations, as well as defining the procedural program flow.
- WS-Addressing–WS-Addressing provides transport-neutral mechanisms to address Web services and messages within an XML format.
- X3D–A standardized XML language used to define 3D computer graphics.
- XBRL–eXtensible Business Reporting Language is an open data standard for financial reporting in XML format.

- XHTML—Is the reformatting of HTML to a XML compliant structure for Web site presentation within Web browsers.
- XPDF—XML version of a PDF document for browser presentation.

The entire repository can be downloaded in a tar.gz file form at <https://www.movesinstitute.org/exi/EXI.html>. A total of 773 XML documents (excluding fragment documents) are evaluated in this thesis to cover a wide range of XML languages as well a wide range of XML file sizes and content with a DoD bias.

F. EXI COMPRESSION STATISTICAL SIGNIFICANCE TEST

While the simple experiment on the DoD categories revealed EXI is the better technique, those experiments did not give a quantifiable measure of statistical significance. To statistically define “better,” two testing methods were used: the Friedman non-parametric and traditional Analysis of Variance (ANOVA) of the four XML compression techniques (DeVore, 2008).

1. Friedman Non-Parametric Test for Randomized Block Experiments

The Friedman test is a non-parametric analysis method for blocked samples of an experiment. In terms of EXI analysis, the blocks are the compression techniques Zip, GZip, EXI schemaless and EXI schema, and the samples are the XML test-corpus documents. The Friedman test is similar to most other comparison analysis techniques in results, but the test requires some unique data reformatting for ranking the results. That is, for each XML document, compression ratios between the four techniques are ranked from 1 to 4 in terms of how well each technique compressed the document. Using the ranked data instead of the raw data eliminates any required assumptions of the distribution of the data and still delivers test of significance.

a. *Friedman Non-Parametric Technique Equations*

As with any other analysis technique, there are both null and alternative hypotheses with the null being rejected based on a test of significance:

- H_0 : There is no difference in effect between block techniques
- H_A : There is a difference

It is important to note is that if the null is rejected, the Friedman test does not indicate where the difference or differences are located. It only indicates the evidence that a difference exist. To find where the differences are located, a multiple comparison technique needs to be applied, such as the Tukey method.

The equation $f_r = \frac{12}{IJ(I+1)} \sum_i R_i^2 - 3J(I+1)$ determines the critical value of the data set and is compared with the Chi-squared $\chi^2_{\alpha, I-1}$ distribution statistic (DeVore, 2008). If the Friedman critical value is greater than the Chi-squared statistic, $f_r \geq \chi^2_{\alpha, I-1}$, then the null hypothesis is rejected and the alternative retained.

- I – The number of blocks.
- J – The number of samples.
- R – The sum of the ranks for block i
- R^2 – The squared sum of ranks for block i

The Tukey method sorts the block averages in increasing order and those that are separated by a factor greater than w , a difference is indicated; $w = Q_{a,m,v} \sqrt{\frac{MSE}{J}}$ (DeVore, 2008).

- Q – is the standardized range distribution
- m – is the number of blocks – 1, that is $(I - 1)$
- v – is the total number of test – I , that is $(I*J - I)$
- J – the number of samples

- MSE – is the Mean Squared Error $MSE = \frac{\sum_{i=1}^I s_i^2}{I} = \frac{SSE}{n - I} = \sigma^2$

$$\circ \quad SSE = \sum_{i=1}^I \sum_{j=1}^J (x_{ij} - \bar{x}_i)^2$$

b. Friedman Non-Parametric Example Demonstration

The example below demonstrates the Friedman technique on four blocking factors on eight samples (DeVore, 2008).

Sample	Block 1	Block 2	Block 3	Block 4
1	22.6	22.5	22.7	23.1
2	53.1	53.7	53.2	57.6
3	8.3	10.8	9.7	10.5
4	21.6	21.1	19.6	23.6
5	13.3	13.7	13.8	11.9
6	37.0	39.2	47.1	54.6
7	14.8	13.7	13.6	21.0
8	14.8	16.3	23.6	20.3

Table 70. Friedman Example Experiment Data set (From DeVore, 2008)

R²	196	361	400	729
R	14	19	20	27
Sample	Block 1	Block 2	Block 3	Block 4
1	2	1	3	4
2	1	3	2	4
3	1	4	2	3
4	3	2	1	4
5	2	3	4	1
6	1	2	3	4
7	3	2	1	4
8	1	2	4	3

Table 71. Friedman Example Experiment RANKED Data set (From DeVore, 2008)

The Friedman equation generates $f_r = \frac{12}{4(8)(5)}(1686) - 3(8)(5) = 6.45$ and at a 0.05 level, the $\chi^2_{0.05,3} = 7.815$. Because $6.45 < 7.815$, that is the Friedman critical value is less than the Chi-squared statistic, the null is retained indicating there is no evidence that any of the blocking factors had any significance compared to the others.

c. EXI Non-Parametric Friedman Analysis

The tested hypotheses for the Friedman analysis on the EXI data set

- H_0 : There is no difference in effect between compression techniques
- H_A : There is a difference

	Zip	GZip	EXI (wo)	EXI (w)
R^2	3896676	2283121	1054729	263169
R	1974	1511	1027	513
Block Rank Average	3.97984	3.04637	2.07056	1.0343

Table 72. Friedman EXI Experiment RANKED Data set Summary

$$f_r = \frac{12}{IJ(I+1)} \sum_i R_i^2 - 3J(I+1) = \frac{12}{4(496)(5)} (7497695) - 3(496)(5) \approx 1629.79$$

and at a 0.05 level, the $\chi_{0.05,3}^2 = 7.815$.

- I – is 4: Zip, GZip, EXI Schemaless, and EXI Schema-informed
- J – 496 schema-informed cases only for the EXI testing. The full 773 cases were not used to ensure an even number of cases per block. Not all XML documents had a supporting schema.
- R – is the sum of the ranks for block i
- R^2 – is the squared sum of block i

Because $f_r \geq \chi_{\alpha, I-1}^2$ is true ($1629.79 \geq 7.815$), the null can be rejected, there is evidence that compression levels depends on which compression technique is applied.

d. EXI Non-Parametric Multiple Comparison Tukey Method

The sorted blocked averages in increasing order for the Tukey's method is listed in Table 73.

EXI (w)	EXI(wo)	GZip	Zip
1.0343	2.07056	3.04637	3.97984

Table 73. EXI Sorted Ranked Averages

The difference “w” level is $w = Q_{\alpha, m, v} \sqrt{\frac{MSE}{J}} \approx 3.63 \sqrt{\frac{0.10135325}{496}} \approx 0.05189$.

- $Q - 3.63$
- $m - 3$
- $v - 1980$
- $J - 496$
- $MSE - \text{is } 0.10135325$

With four blocks there are a total of $4 \text{ choose } 2 = 6$ possible combinations of differences listed in Table 74.

Comparison	Difference
Zip-Gzip	0.933
Zip-EXI(wo)	1.909
Zip-EXI(W)	2.946
Gzip-EXI(wo)	0.976
Gzip-EXI(w)	2.0121
EXI(wo)-EXI(w)	1.036

Table 74. EXI Block Averages Differences

e. Conclusion of EXI Non-Parametric Friedman Analysis

Because the Friedman critical is greater than the Chi-squared statistic the null hypothesis that there is no difference between the techniques can be rejected.

The Tukey's multiple comparison shows that all six of the differences are greater than "w" indicating that each technique is statistically different from the others, and because the EXI technique delivered the smallest average, it is the superior XML compression technique.

2. Analysis of Variance (ANOVA)

An ANOVA is a statistical analysis technique similar to Friedman's except that it requires the data to be have independence and be normally distributed, and have equal variance within each block (compression technique). As shown in the following modeling sections, these assumptions can be made and are supported by the large sample size enabling the Central Limit theorem. The Central Limit theorem states that as the number of samples from a distribution whether known or unknown with a mean μ and variance σ^2 , the distribution of the samples approximates a normal distribution demonstrated in Figure 65. Simply, the larger the value of n , the better the approximation (Sanchez, 2009).

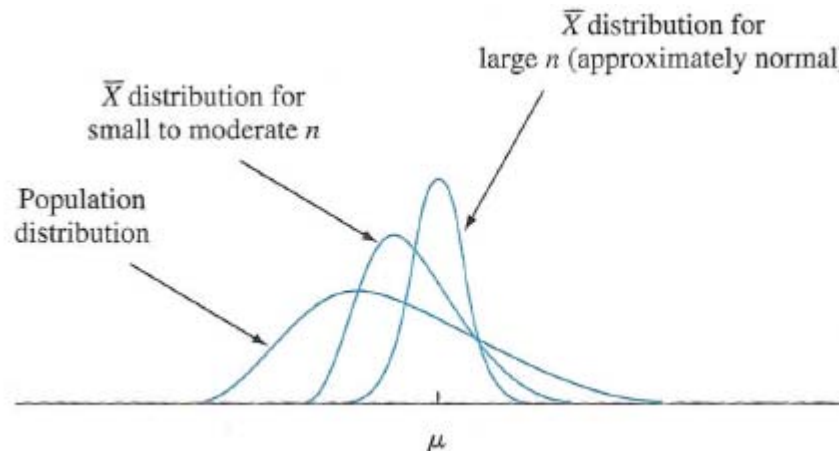


Figure 65. Example of the Central Limit Theorem (From Sanchez, 2009)

The concept of an ANOVA is the testing of the assumption all of the blocks estimate the same σ and does so by pooling the data of the groups (DeVore, 2008).

$$\frac{\text{variance between blocks}}{\text{variance within each block}} = \frac{\text{explained}}{\text{unexplained}} = \frac{MST}{MSE} = 1 \text{ if null is true}$$

a. ANOVA Equations

As with any other analysis technique, there are both null and alternative hypotheses with the null being rejected based on a test of significance:

- H_0 : There is no difference in means between block techniques
- H_A : There is a difference

By pooling the variances within each treatment group an estimate of σ^2 is derived, traditionally called the Mean Square Error (MSE), the pooled estimate of the error variance. The Sum of Squares of Errors (SSE) is how much variation exist within each block. The Sum of Squares (SST) within a block is the variance of the block's data, and the Sum of Squares of the Sum of Squared residuals (SSTr) is the variance of the data between the blocks that can be explained by possible differences in the mean and standard deviations (DeVore, 2008).

The set of equations to derive ANOVA results follows:

- I = the number of populations or treatments being compared
- J = the number of observations within each treatment
- n = total number of samples (incase not same J observation for each of I treatments)
- \bar{x}_i is the mean of a block
- $\bar{\bar{x}}$ is the grand mean of the entire data set
- s or σ is the standard deviation and s^2 or σ^2 is the variance

$$\bar{x}_i = \frac{\sum_{j=1}^J x_{ij}}{J} \quad i = 1, 2, 3, \dots, I \quad \bar{\bar{x}} = \frac{\sum_{i=1}^I \sum_{j=1}^J x_{ij}}{n} \quad s_i^2 = \frac{\sum_{j=1}^J (x_{ij} - \bar{x}_i)^2}{J - I}$$

$$SST = \sum_{i=1}^I \sum_{j=1}^{J_i} (x_{ij} - \bar{\bar{x}})^2 = \sum_{i=1}^I \sum_{j=1}^{J_i} x_{ij}^2 - \frac{1}{n} \bar{\bar{x}}^2 \quad df = n - 1$$

$$SSTr = \sum_{i=1}^I \sum_{j=1}^{J_i} (\bar{x}_i - \bar{\bar{x}})^2 = \sum_{i=1}^I \frac{1}{J_i} x_i^2 - \frac{1}{n} \bar{\bar{x}}^2 \quad df = I - 1$$

$$SSE = \sum_{i=1}^I \sum_{j=1}^{J_i} (x_{ij} - \bar{x}_i)^2 = SST - SSTr \quad df = \sum_{i=1}^I (J_i - 1) = n - I$$

$$SST = SSTr + SSE$$

$$MSTr = \frac{J}{I - 1} \sum_{i=1}^I (\bar{x}_i - \bar{\bar{x}})^2 = \frac{SSTr}{I - 1}$$

$$MSE = \frac{\sum_{i=1}^I s_i^2}{I} = \frac{SSE}{n - I} = \sigma^2$$

$$f = \frac{MSTr}{MSE} \text{ test statistic}$$

A test of significance like Friedman's test does not indicate where the difference can be found, just that there is a difference between at least two blocks:

- H_0 retained if $E(MSTr) = E(MSE) = \sigma^2$ or $f = \frac{MSTr}{MSE} = 1$
- H_0 rejected if $E(MSTr) > E(MSE) = \sigma^2$ or $f \geq F_{\alpha, I-1, n-I}$

To find where the differences are located, a multiple comparison technique needs to be applied, such as the Tukey method.

<i>ANOVA Table</i>				
Source of Variation	DF	Sum of Squares	Mean Square	<i>f</i>
Treatments	I - 1	SSTr	$MSTr = \frac{SSTr}{I - 1}$	$\frac{MSTr}{MSE}$
Error	n - I	SSE	$MSE = \frac{SSE}{n - I}$	
Total	n - 1	SST		

Table 75. ANOVA Calculation Table (After DeVore, 2008)

b. Conclusion of EXI ANOVA

The measure of performance for the ANOVA is the compression ratio that the technique produced (technique/original). Using this measure normalizes all blocked effects for each document regardless of original file size, number of elements, attributes and values contained within each document thereby generating an equitable comparative measure between techniques.

The results of the ANOVA analysis concludes that EXI schema-informed and EXI schemaless generate statistically significant smaller files compared to all other techniques at a 95% confidence level as shown in Figure 66. Significance is denoted by visual inspection of the Tukey-Kramer circles and confidence interval diamonds, noting that the circles and diamonds do not overlap with any other diamond or circle. Overlapping circles or diamonds would indicate a lack of statistical significance. Since EXI does not overlap any other techniques and is below the other techniques, it indicates EXI is statistically superior. Given the measure of performance is percentage of the original document and that EXI's average was lower, more compressed than the other techniques, EXI both schema-informed and schemaless are the superior XML compression techniques.

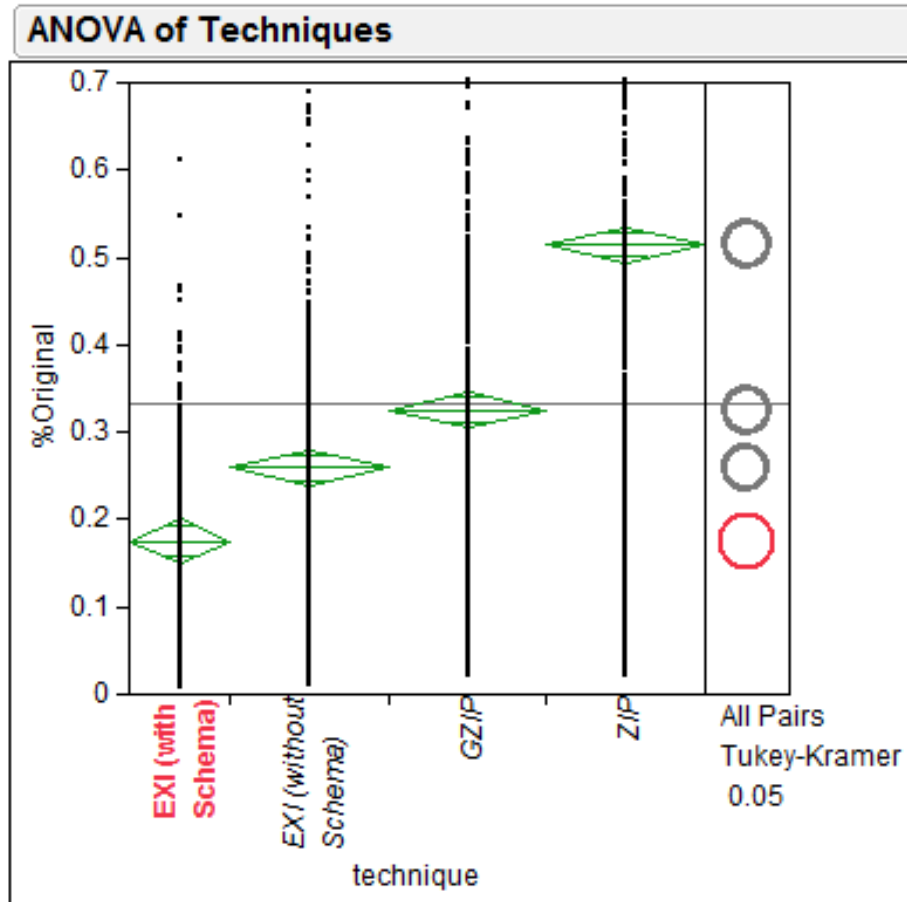


Figure 66. Analysis of Variance (ANOVA) Comparison of XML Techniques

Note that in Figure 66, the % Original range was truncated to the maximum range of the EXI results for clarity. Both Zip and GZip had occurrences that exceeded 100% of the original XML document and in the worst case 400%. The key points of this plot are the confidence interval diamonds, not the full range of the results.

For additional clarity and alternative perspective of the significance between techniques, a tabular representation of the ANOVA significance is included in Table 76 for the confidence interval (diamonds) and Table 77 for the test of significance (Tukey-Kramer circles). Within Table 76, by noting that confidence interval ranges between EXI and the other techniques do not overlap indicates EXI is statistically significant, and in this case, statistically superior. Within Table 77, a similar overlap

comparison is done, but using the lettered columns. Each technique is assigned a column and each technique's range is assigned a letter. Any techniques that have overlapping ranges would have multiple letters assigned. The lack of multiple letters per technique row indicates significance between techniques.

Encoding	Number	Mean	Standard Error	Lower 95%	Upper 95%
EXI (with Schema)	496	0.175	0.01278	0.15	0.2
EXI (without Schema)	775	0.259	0.01022	0.239	0.279
GZIP	775	0.325	0.01022	0.30523	0.345
ZIP	775	0.515	0.01022	0.495	0.535

Table 76. Moments of the ANOVA Comparison of XML Techniques

Encoding	Level Alpha Means				Mean
ZIP	A				0.515
GZIP		B			0.325
EXI (without Schema)			C		0.259
EXI (with Schema)				D	0.175

Table 77. Tukey-Kramer Statistical Significance Difference between Techniques Measurement

3. Conclusion of Test of Significance Between Compression Techniques

Both the parametric ANOVA and non-parametric Friedman test highlighted that EXI is the best XML compression technique and that all the techniques are statistically different from one another.

It is important to realize that in no case did either of the EXI techniques deliver a result file equal to or greater than the original input XML document. Both GZip and Zip had numerous cases where they exceeded the original document, and in the worst case, exceeding the original document by 400%. The distribution of the result of the techniques applied to the repository of test cases is included in Figure 67. The red arrows are present to highlight the original document mark for the techniques. All results above the red arrows are compression cases that resulting is a file larger than the original file.

Because EXI always delivered results smaller than the original document, the original document mark arrow is not included for the EXI techniques.

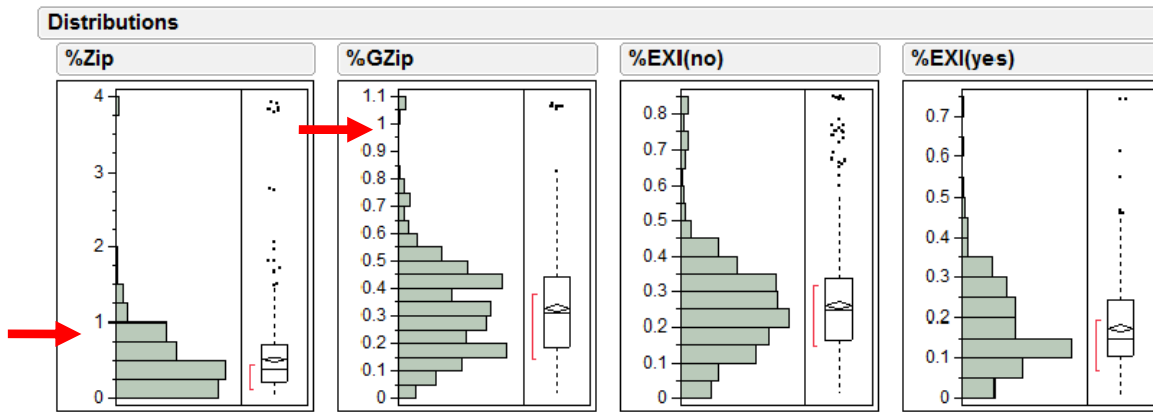


Figure 67. Distribution of Compared XML Technique–Percentage of Original Document

G. STATISTICAL PREDICTIVE MODELS

As a tool to test the predicted effect of EXI within a prospective XML domain, statistical models are presented based on the test-corpus of documents. Only EXI schemaless prediction models are developed. This is justified in that not all XML documents within any domain have a corresponding schema, nor are all the XML documents within any domain assured to follow the schema precisely. For those XML documents that do have a schema and follow the schema precisely, EXI schema-informed processing always produces results that are more compact than EXI schemaless. Schemaless EXI is thus a performance upper bound for the EXI technique. Schemaless is a more general and universally applicable approach to modeling EXI effectiveness.

1. Model Factors of Interest

Based on the innate structure of XML, Table 78 defines the factors captured for each sample case from the test-corpus of XML document to build the statistical models.

#	Factor (Factor ID)	Factor Description
1	Maximum Depth (MD)	<ul style="list-style-type: none"> This factor defines the maximum depth of the XML structure. This factor captures the effect of repeating element (grammars).
2	Number of Unique Elements (NE)	<ul style="list-style-type: none"> This factor counts the number of unique elements within the XML sample document. Considering the EXI algorithm, elements of an XML document defines the number of grammars and indicated the number of possible transitions and unique values.
3	Count of Elements (CE)	<ul style="list-style-type: none"> This factor is the raw count of elements within the XML document. This factor provides a repletion ratio of grammar usage within the XML documents: $\text{count} / \text{unique} = \text{average replication of grammars}$
4	Number Unique Attributes (NA)	<ul style="list-style-type: none"> This factor counts the number of unique attributes within the XML document. Attributes are always followed by a value and in terms of EXI equates to a string table entry and EXI stream raw ASCII entry.
5	Count of Attributes (CA)	<ul style="list-style-type: none"> This factor is the raw count of attributes within the XML document. This factor indicates grammar attribute replication and possible value replication capture.
6	Number of Unique Value fields (NV)	<ul style="list-style-type: none"> This factor is the count of the number of unique values found within the XML document sample (attribute value and element content). This factor is an indicator of string table sizes.
7	Count of Value Fields (CV)	<ul style="list-style-type: none"> This factor is the raw count of value fields including duplicate values. This factor provides an indication of values replication ratio.
8	Content Density (CD)	<ul style="list-style-type: none"> This is the sum of character value lengths / document size This factor captures some of XML's unavoidable innate variability.

Table 78. XML Descriptive Factors Captured for EXI Prediction Modeling

2. Sampled Data Assumptions and Mitigation for Modeling

The baseline requirements needed for a valid parametric regression model (DeVore, 2008):

a. Independence of the observation in the sample.

Statistical independence is met based on the method used to derive the sample XML documents for the test-corpus. Random documents from various XML families of languages are utilized for inclusion in the XML sample population, i.e., the EXI test-corpus.

This condition is verified by scatter plot of the residuals by the predicted values ensuring no clustering, outliers or curves are present in the plot. Presence of these items can indicate the samples are not independent.

b. Linearity of the expected value as a function of the independent variables

Linearity will be verified through two scatter plots: Actual by Predicted, and Residuals by Predicted. The plot points if linearity is achieved will be symmetrically distributed around a diagonal line for the Actual by Predicted plot and a horizontal line for the Residuals by Predicted plot.

If the plots do not indicate linearity, data transformations on the dependent variable or the independent variable can be performed. Often right skewed data, sometime called strictly positive, can be transformed with a logarithm to achieve linearity. Additionally, quadratic terms can be added, that is, the product of several predictor variables can be used to compensate for quadratic shapes.

c. Equal variance of the errors of the dependent variable

Equal variance is checked by a scatter plot of the Residuals by Predicted values with a symmetric distribution of the plot points. When the Residuals by Predicted plot indicates a lack of equal variance, a plot of the Residuals by some of the predictor variables can indicate which predictor variables are injecting the most variance into the model.

Variance is not easily removed, and generally can only be resolved by increasing the sample size or the removal of independent variables that are injecting the variance. Neither of these are easy solutions. Often increasing the sample size cannot be increased and the removal of independent variables degrades the performance of a model.

d. Normally distributed dependent variable

Normality the dependent variable is checked with a histogram of the residuals and visually inspecting looking for a uni-modal and uniform distribution shape. If this method does not provide the necessary confidence, a normal probability plot will be used.

A normal probability plot sorts the data from lowest to highest and assigns a percentage to each data using $\frac{(i-.5)}{n}$ where i is the i^{th} sorted data point and n is the total number of data points. The corresponding normal z-score for the percentage is listed for each point. The graph is constructed using the data and z-scores as the upper and lower bounds of the x-axis and y-axis with the origin of the graph as a y value just under the lowest observation and a slightly lower z-score for the x value. The plotted point coordinates are (observed, z-score). The closer the plot makes a straight line, the more normal the data. Any s-shape pattern indicates a lack of normality in the data.

If the data does not appear to be normally distributed, then it indicates that the distribution may not be normal and or that the linearity assumption may be in violation. To overcome a lack of normality, data transformations on either or both the

dependent variable or the predictor variable may transform the data into a more normal distribution. Another method may be to eliminate the individual data points that are the cause of the lack of normality. However, this option is only applicable if individual justifications can be made for their removal such as errors in the data collection, extremely rare occurrences, or other reasonable justification for ignoring their input.

e. Data Transformation Consideration–Dependent Variable

The initial model building indicates a strong lack of linearity, uncertainty of equal variance and a lack of normally distributed data. To compensate for these findings, data transformations are applied to the dependent variable to convert it into a form that is consistent with the required assumptions needed to continue with model generation. Several transformations were applied as listed in Table 79, with each resulting transformation distribution shown in Figure 68. A quick review of natural logs:

- $\ln(0) = NaN$, all logs must have positive value input. Zero is not negative nor is it positive.
- $\ln(.5) \approx -0.69$ as the input approaches 0, the result becomes more negative.
- $\ln(1) = 0$ $10^x = 1$ $x=0$
- $\ln(2) \approx 0.69$ as the input increases to positive infinity, the results increase positively.

The Natural Logarithm	The Inverse	The Inverse Square
$y_i' = \ln(y_i)$	$y_i' = \left(\frac{1}{y_i}\right)$	$y_i' = \left(\frac{1}{y_i}\right)^2$

Table 79. Transformation Algorithms Attempted on EXI Results

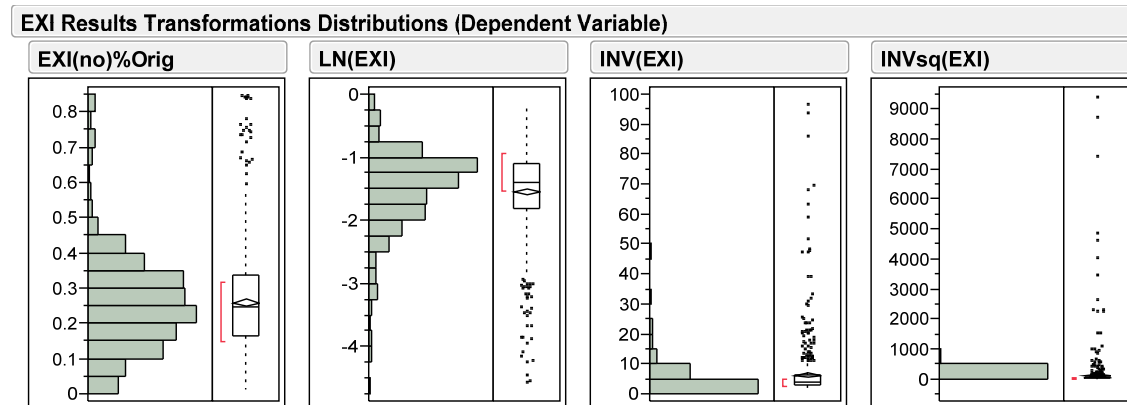


Figure 68. EXI Restuls Distributions (Transformed)

Based on the transformed EXI distributions, the raw and natural logarithm results deliver the best looking normal like distributions, with the natural logarithm having the comparative best based on its lower variance. This is as would be expected with the skew of the original result's distribution; heavily right-skewed data generally benefits from a logarithm function. Although the natural logarithm transformed distribution is still not ideal, it takes on better distribution characteristics.

Using the natural logarithm transformation, the nearly normal assumption can be made based on the shape of the histogram and the narrower variance, although there still remain numerous outliers. The assumption of normality and equal variance is a stretch, but must be made in order to pursue a parametric model for EXI performance prediction.

f. Data Transformation Considerations–Independent Predictor Variables

Continued model building still left normality, equal variance and most importantly linearity characteristic is less than optimal as shown in Figure 69. To accommodate this, the same transformation used on the dependent variable are applied to the independent predictor variables. Most important to note after the transformations are applied is the increase in linear association achieved between the factors and the dependent variable, EXI results, as shown in Figure 70. Although some factors still have room for improvement, compared to the untransformed data the linearity associations are

clearly present though with quite a bit of variance. A linear-regression model requires a linear association of factors to result, and these transformations have delivered the linearity requirement.

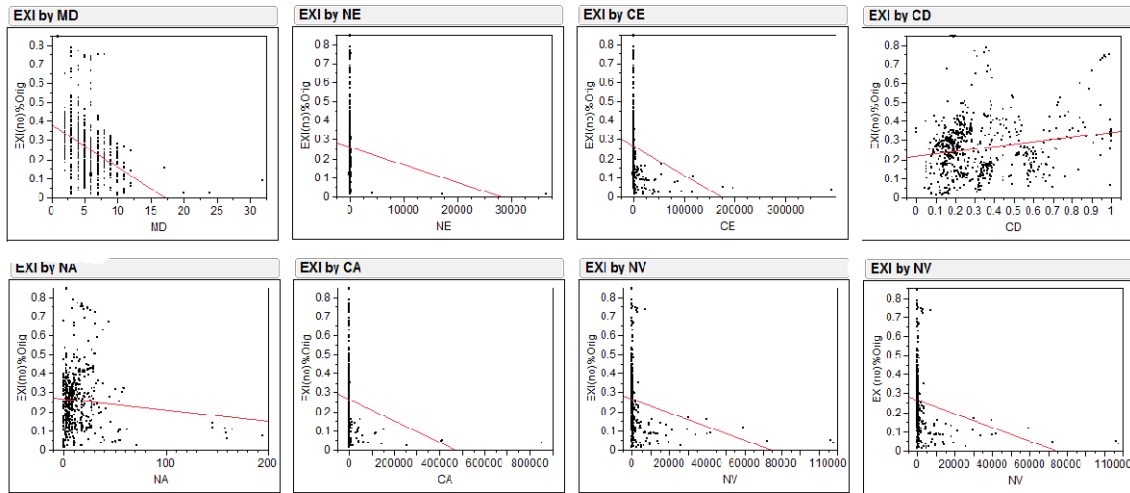


Figure 69. Predictor Variables by EXI Schemaless Results (Untransformed)

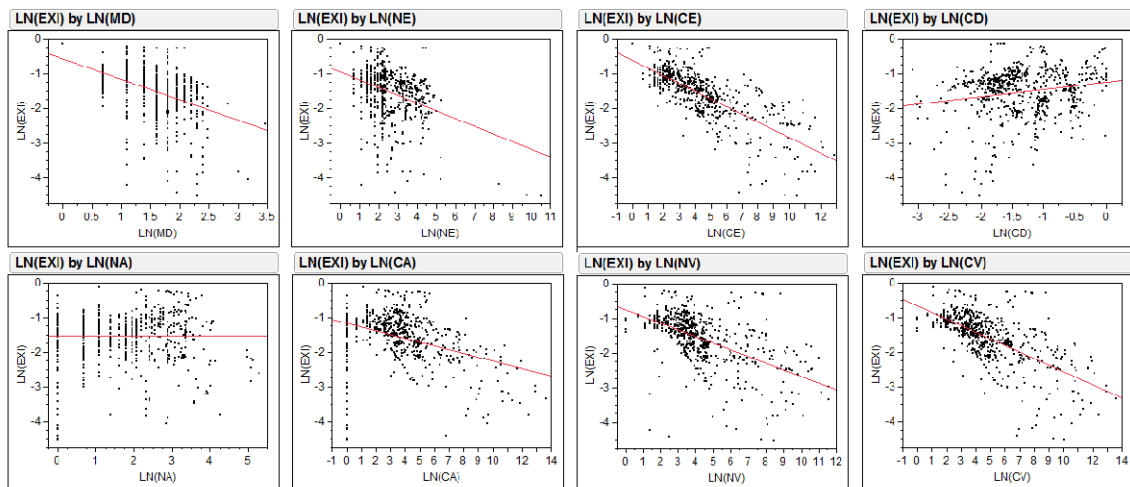


Figure 70. Predictor Variables by EXI Schemaless Results (Transformed)

3. Variance Between and Within XML Documents

a. *The Source of Variance*

The source of variance in XML documents can be found in their value content, or content density. Two documents with the same factor settings can generate uniquely different compression ratios due to differences in content alone. For example, the XML documents contained in Table 80 (small) and Table 81 (larger) both have the same factor levels, other than content density, but the resulting compression ratios are different as shown in Table 82 when the different techniques are applied.

b. *Example of Variance*

Both of the example documents' factor settings:

- Depth (MD): 3
- Number of Unique Elements (NE): 3
- Count of Elements (CE): 3
- Number of Unique Attributes (NA): 2
- Count of Attributes (CA): 2
- Number of Unique Values (NV): 3
- Count of Values (CV): 3

```
<?xml version="1.0" encoding="UTF-8"?>
<orders>
  <order orderBy="a customer short name" orderDate="someDate">
    <productDescription>A short description of what
      was ordered</productDescription>
  </order>
</orders>
```

Table 80. XML Document Variance Example (Small)

```
<?xml version="1.0" encoding="UTF-8"?>
<orders>
  <order orderBy="a customer with a longer name
    than the short name" orderDate="someDate">
    <productDescription>This product order description
      is a longer and more detailed description,
      though has same XML structure as the shorter
      version of orders</productDescription>
    </order>
  </orders>
```

Table 81. XML Document Variance Example (Large)

Document	Technique	Original Size	Compressed Size	% of Original
varianceExampleSmall.xml	GZipped	229	167	72.93%
	Zipped		473	206.55%
	EXI		106	46.29%
varianceExampleLarger1.xml	GZipped	372	231	62.1%
	Zipped		541	145.43%
	EXI		175	47.04%

Table 82. XML Document Variance Experiment Test Results

Both of the example documents have identical structures (factor settings), but different compression ratios for all techniques due to the differences in document content, the content density (CD) factor in the models. This equates to XML documents that have a wide variance from document to document, within and between various XML families of languages.

While the EXI difference in this simple case is only 1%, as the documents are enlarged with many more elements and attributes, the delta in compression ratios increases. A 1% difference in such small document highlights the diverging potential of XML document variance: $Var(x) \neq c \quad \forall x \in XML$; XML does not have equal variance.

c. Variance Mitigation Techniques

The model factor named Content Density (CD) attempts to capture XML's source of variance by using the document's ratio of data values to XML structure as a factor input to the model. This is a best-effort attempt, and is not a perfect solution because it does not capture repeating values, which are a major driving factor of

compression performance. Indirectly the ratio of repeating values is approximated by CV over NV, the count of values including repeats divided by the number of unique values. As this ratio increases (greater than 1), it indicates the larger portion of repeating values within the XML document. Both of CV and NV are input factors when building EXI performance-prediction models.

Quite simply, XML documents have variance, and that variance cannot be precisely mitigated or generalized. However, as indicated by the CD and CV/NV, measurements of approximate variance can be taken and marginalized within statistical models.

4. Parametric Prediction Model

Parametric models such as a Least Squared Error (LSE) are linear models based on the coefficients of the parameters, and often generate very accurate and simple models that can be easily implemented. LSE model are often the initial choice when modeling data for their accuracy and simplicity.

a. *Parametric Models in General*

Parametric regression models are error based, also called residual models, $\sum (observed - model)^2 = \min_{(m,b)} \sum (y_i - mx_i - b)^2$ (Hand, Manilla & Smyth, 2001 & Hastie, Tibshirani & Friedman, 2009). These models operate on the assumption that all errors are identically and independently distributed (IID) and that the assumptions listed in the previous sections are met. Generally, these types of models are good in that they are simple and produce a good fit of points. However, the Achilles heel of parametric model's is that they consider all points with equal predictive weight including points of noise, outliers as well as genuinely good values. Large amount of noise or outliers will negatively affect a parametric model's ability to predict accurately.

Methods to try to alleviate parametric shortfalls are to transform the data with logarithms or other functions in an attempt to eliminate the impact of data skew on the model (DeVore, 2008). Additionally, a custom weighting function can be added that

attempts to minimize the effect of outliers on the model using a distance-like function.

An example function could be $\frac{d^2}{(d^2 + s^2)}$ with d = a distance or error, and s = some

dissimilarity metric to decrease the impact of the outlier that drives the function to zero as the error increases (Hand, Manilla & Smyth, 2001 & Hastie, Tibshirani & Friedman, 2009) Roughly speaking, this approach enables the model to intelligently throw out some questionable observations by assuming they are out of the range of valid observations.

b. Parametric Model Measure of Variance

Using the transformed data set as defined in the previous section, the parametric regression model achieved an adjusted R-squared of 0.863 as listed in Table 83, indicating that over 86% of the variance in the data is accounted for in the model. Given the skew of the EXI results data, lack of normality and lack of linearity in the original data, this is a relatively good model. Indeed it exceeds initial expectations in a parametric model's ability to predict the expected performance of EXI.

Model Characteristic	Value
R-Squared	0.864949
R-Squared Adjusted	0.862636
Root Mean Square Error	0.250826
Mean of Response	-1.52886
Observations (or Sum Wgts)	773

Table 83. Summary of Parametric Model Fit (Transformed Data)

The following is a quick review of R-squared adjusted and R-squared interpretations to solidify the impact of such a score given the shape of the original EXI results data. They both measure in range [0, 1] the amount of variability (how accurate) of the model to the underlying data; the closer to 1 the better (DeVore, 2008). Ideally, a perfect model will have no residuals (actual – predicted = 0), that is, the model perfectly matches the provided data with an equation so that a line can be drawn directly through every observation without missing any, e.g., R-squared = 1. The R-squared metric is used as a measurement of how close the model is to being perfect.

R-Squared is calculated with the Residual Sum of Squares (RSS) method, which is the difference between the actual observation and the predicted observation, divided by the Total Sum of Squares (TSS), which is how the actual observations differ from their mean (Hand, Manilla & Smyth, 2001 & Hastie, Tibshirani & Friedman, 2009).

$$R^2 = \frac{RSS}{TSS} = \frac{Model}{Data} \quad RSS = \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad TSS = \sum_{i=1}^n (y_i - \bar{y})^2$$

For models that have multiple factors, such as the EXI models, the R-squared metric is artificially driven up with each added factor because each factor contributes to the model effect additively, but not always significantly (Hand, Manilla & Smyth, 2001 & Hastie, Tibshirani & Friedman, 2009). The adjusted R-Squared accounts for added factors and only increases if the residuals are reduced.

$$R_{Adj}^2 = 1 - \frac{\frac{RSS}{n-1}}{\frac{TSS}{n-1}} = \frac{(n-1)R^2 - k}{n-1-k} \quad n = \# \text{ observations, } k = \# \text{ factors}$$

c. *Parametric Model Equation*

The parametric model as listed in Table 84 found only five of the eight factors in and of themselves as significant. All eight factors are used in the model, but three of the factors are only found in interactions (MD, NA, CV) and polynomial effects (NA, CV). Note that factors when within interactions or polynomial effect are simply products of the two, and that they use their mean centered value rather than the raw factor value. The chief advantage of the centering is that it reduces multicollinearity or high correlation between the factors (DeVore, 2008).

Term	Coefficient Estimate
Intercept	-0.52628
LN(NE)	0.2814966
LN(CE)	-0.4995
(LN(NA)-1.70991)*(LN(NA)-1.70991)	-0.045213
LN(CA)	-0.086833
LN(NV)	0.2434582
(LN(CV)-4.51785)*(LN(CV)-4.51785)	-0.045667
LN(CD)	0.3126409
(LN(MD)-1.60724)*(LN(NE)-2.58374)	0.0843596
(LN(NE)-2.58374)*(LN(NA)-1.70991)	0.0919068
(LN(NE)-2.58374)*(LN(CV)-4.51785)	-0.060353
(LN(CE)-4.11961)*(LN(NV)-4.00658)	0.0714633
(LN(CA)-3.40475)*(LN(CV)-4.51785)	0.0146555
(LN(NV)-4.00658)*(LN(CD)+1.333)	0.1054722

Table 84. Parametric Model Parameter Estimates (Transformed Data)

d. Parametric Model Usage Characteristics

Some important characteristics to remember about this model are both the factors and model results must be transformed before being useable by the model or interpretable as equivalent result units. Table 85 lists some transformation examples of the model predicted results, noting that a negative untransformed model prediction value equates to better compression than a positive value; the more negative the model result the increased the predicted compression.

- Predictor variables must be transformed with a natural logarithm, $\ln(x_i)$.
- Model predictions must be transformed with an exponent, $e^{\text{Model Results}}$.
- Model results range $-\infty \leq \text{model} \leq 0$
 - 0 = no compression or 100% of the original file
 - $\lim_{\text{Model Result} \rightarrow -\infty} = \text{better compression}$

Actual x	Model Results $Ln(x)$	Original Units e^x
0.01	-4.60517	0.01
0.02	-3.91202	0.02
0.03	-3.50656	0.03
0.04	-3.21888	0.04
0.05	-2.99573	0.05
0.1	-2.30259	0.1
0.2	-1.60944	0.2
0.3	-1.20397	0.3
0.4	-0.91629	0.4
0.5	-0.69315	0.5

Table 85. Parametric Model Results Transformation Examples

For example, if a sample XML document had the following factor characteristics, the model result is -2.22713634268882:

- (MD) Depth = 6
- (NE) Number of Unique Elements = 127
- (CE) Count of Elements = 229
- (NA) Number of Unique Attributes = 2
- (CA) Count of Attributes = 4
- (NV) Number of Unique Values = 78
- (CV) Count of Values = 152
- (CD) Content Density = 0.04900332

The model result transformed $e^{-2.22713634268882} = 0.107836796027303$ which is the original ratio units of desire; a predicted compression ratio of the sampled XML document of 10.8%. Key to remember is that the more negative a number from the model before being transformed by the exponent, the smaller the resulting compression ratio (Table 85), e.g., the more compact.

e. Parametric Model Parameter Interpretation

The interpretation of the factors coefficients in their ability to estimate the effects of EXI performance are defined as follows.

(1) Intercept. For this model the intercept has no genuine meaning. Traditionally the intercept equates to the prediction results if all factors are zero. In the case of all factors 0, the input XML documents would be invalid and EXI becomes not applicable. However, the intercept could be thought of a basic overhead associated with all XML documents.

(2) Maximum Depth (MD). The depth of the XML document had no standalone factor effect on the predictability of EXI performance, though did show up as an interaction with the Number of Unique Elements (NE).

(3) Number of Unique Elements (NE). Based on its positive coefficient value, NE negatively effects compression. As the number of unique elements within the XML document increase, the predicted level of compression is reduced. This result is reasonable given that the more elements are present in a XML document, the more raw ASCII text that has to be coded into the EXI stream, which equals reduced compression from the higher quantities of raw ASCII. NE also impacts the bit-size of element tokens. NE is also found as an interaction between NA, CV and MD.

(4) Count of Elements (CE). The negative coefficient highlights that as the total number of elements within an XML document increases, increased compression gains are predicted. This is likely due to the fact that as the raw number of elements, both unique and duplicates increase, the likelihood of replication increases. This is a reasonable result. CE is also found as an interaction with NV.

(5) Number of Unique Attributes (NA). The number of unique attributes did not come up as a main effect, but was both a polynomial ($NA*NA$) and an interaction with NE. Because the polynomial effect is negative, as the level of NA deviates from its mean value (transformed value of 1.709), the compression level improves.

(6) Count of Attributes (CA). Similar to the count of elements (CE), the negative coefficient equates to better expected compression ratios as the number of attributes increase. The implied explanation is that attributes often contain repetitive data and so as the number of attributes increases, the repetition probability increases, though not at the same rate as elements as indicated by the lower coefficient compared to the count of elements. Again this is a reasonable result.

(7) Number of Unique Values (NV). This factor, other than the number of elements (NE), has the most impact on the model's prediction. Based on the positive coefficient, as the number of unique values increases, predicted compression level decreases. Intuitively, as the number of values within an XML document increase, compression will decrease because each unique value has to be coded as a raw ASCII value on its first occurrence; the more ASCII then the lower the compression potential.

(8) Count of Values (CV). The count of values did not come up as a main effect, but was both a polynomial ($CV*CV$) and an interaction with NE and CA. Same as the NA polynomial, the further the level of CV deviates from its mean value (transformed value of 4.51785), the predicted compression level improves. The implied meaning behind this is that as the total number of values increases, the likelihood that some of the values are repeating values increases. This interpretation is based solely on the known structure of XML and is not statistically provable, but is reasonable.

(9) Content Density (CD). The final factor CD did come up as a main effect and with a coefficient that might be expected. CD has a positive coefficient or negative effect on compression. As the content density of the document increases, the compression effect of EXI decreases. That is, the more the document consist of values, the lower the compression EXI can likely deliver; the more values in a document, the less likely it will contain exact value repetition.

(10) MD and NE Interaction. This interaction has a positive coefficient or a negative effect on compression as the MD and or NE increases. This is reasonable given an increase in either result in a new Grammar with basic overhead associated, and that also implies certain limits on compression such as the ASCII coding of the element name.

(11) NE and NA Interactions. This interaction has a positive coefficient or a negative effect on compression as the NE and or NA increases. This, like the MD * NE interaction is reasonable given the basic overhead of each element during its first occurrences.

(12) NE and CV Interactions. This interaction has a negative coefficient or a positive effect on compression as the NE and or CV increases. This is reasonably explained as NE is constant and CV increases, the likelihood of repeated values increases. The larger the number of repeating values, the better the EXI compression results.

(13) CE and NV Interactions. This interaction has a positive coefficient or a negative effect on compression as the CE and or NV increases. As CE increases, the likelihood of repeated element content values decreases given in general element content is unique between elements. Repeating values in the XML family of languages is generally found within attributes and not elements.

(14) CA and CV Interactions. This interaction has a positive coefficient, which is a negative effect on compression as the CA and or CV increases. Initial review of this term is puzzling, but after further consideration the results are likely the effect of being a linear combination of each other; essentially, they are the same measure. For either of the factors as a standalone, their increase degrades compression due to overhead for CA and first-time occurrence ASCII encoding for both.

(15) NV and CD Interactions. This interaction has a positive coefficient or a negative effect on compression as the NV and/or CD increases. This interaction is an indication of the level of replication within the XML document. Content Density (CD) being the percentage of the document that is values, then as the number of values (NV) within the document increases, the amount of repeating values decreases; compression performance decreases.

f. Parametric Model Factor Profile

A pictorial example of how each factor affects the model is contained in Figure 71. The direction of the slope of the blue line as the factor level changes across the x-axis indicates the affect of that factor on the model's result indicated on the y-axis. Remember, the more negative a result is the better compression predicted. Visual inspection shows that as CE, CA and CV increase in value (move right) they have the most impact on compression improvement, and that as NE, NV and CD decrease (move left) in value have the most impact on compression degradation.

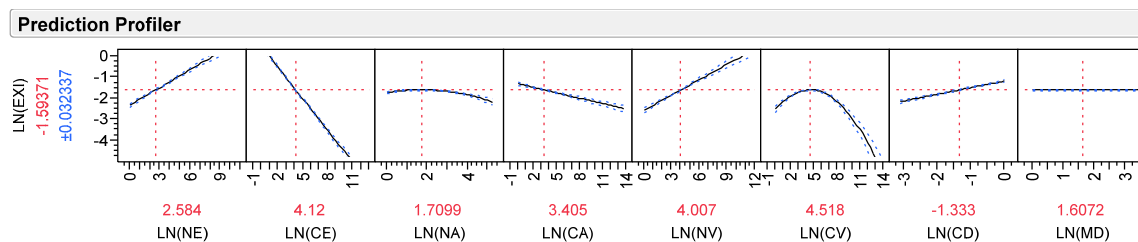


Figure 71. Parametric Model Factor Profiles, How One Unit Change in a Factor Effects the EXI Predicted Results

g. Parametric Model Factor Impact on Model

The impact of each term of the model is depicted in Figure 72, sorted in decreasing order of impact. The red bars indicate the impact level, standardized between the terms. The further the red bar is from the center, the more impact it has on the model. Also, the direction which the red bar protrudes from the center indicates the signed impact of that factor on the model, since those bars that protrude left impact the model for improved compression predictions and those that protrude to the right reduce the predicted compression level. The blue line is the threshold for term significance. A term is only significant if the red bar exceeds the blue line; all terms of this model are significant.

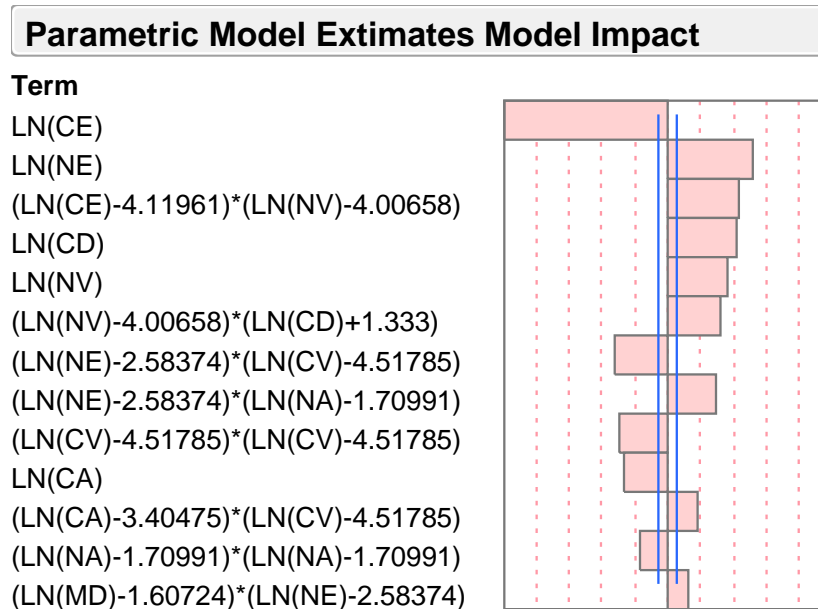


Figure 72. Parametric Model Terms Impact on the Compression Results

h. Parametric Model Analysis of Fit and Feasibility

The Actual by Predicted plot shown in Figure 73 indicates the quality of the fit of the model is good, but not perfect. This plot is basically a pictorial representation of the adjusted R-square value. Ideally, the closer the plots form a straight line, overlying the red line, the better the prediction accuracy of the model. Based on visual inspection of the plot, this EXI prediction model does show a good linear fit without bends or curves, but does have some variation in the distribution of the points around the linear flow. However, the variation is small so linearity assumptions are met.

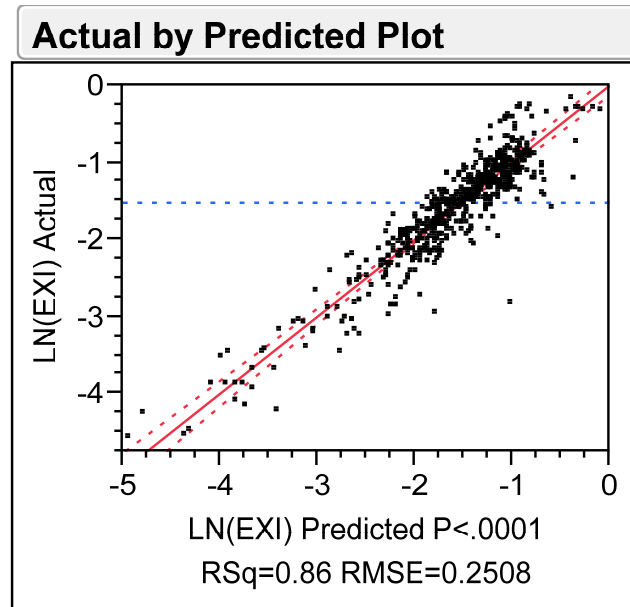


Figure 73. Parametric Model Actual by Predicted Plot

One of the most important plots is the Residual by Predicted plot shown in Figure 74. The indicators of a bad model are patterns or bends in the distribution of the plot points. The EXI prediction model results however are relatively good. The distribution above and below the y-axis zero is reasonably symmetric in terms of maximum and minimum values and that range tends to be found across the range of the plot's x-axis. The clustering of the data around the x-axis values between -2 and -1 is not of concern, it is just how the data was predicted by the model. The true indicators are the symmetry and lack of noticeable pattern in the plot of points. For real-world data, this is a good plot and indicates a good fit of the model to the data: independence, linearity and equal variance assumptions met.

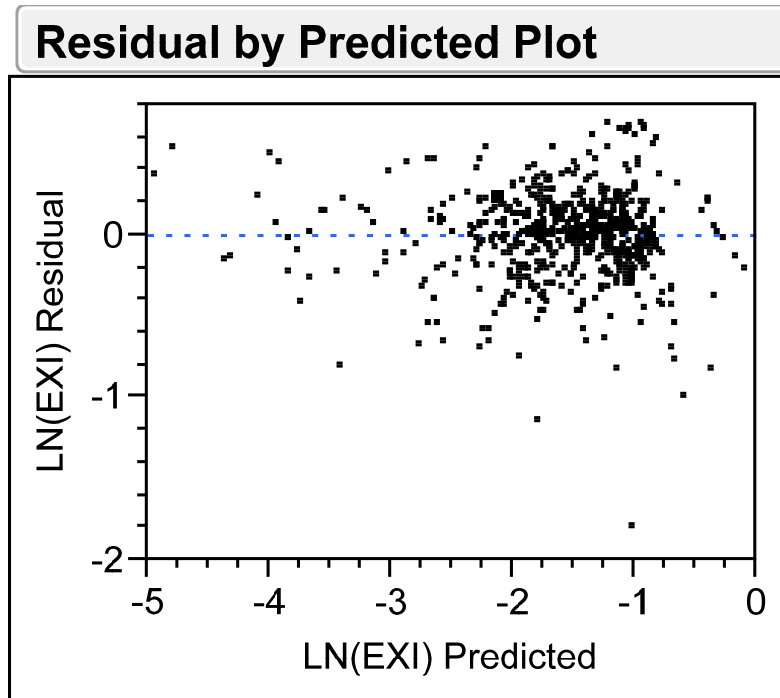


Figure 74. Parametric Model Plot of Residuals by Predicted Value

Figure 75 is a plot of the residual values distribution, that is (actual minus predicted) and both Table 86 and Table 87 are the statistics of the distribution: a mean of 0.7% and standard deviation of 7% (standard error of 0.2%) of the predicted value to the actual result. The shape of this distribution is unimodal and normal and for real world data is a great example of a normal distribution; normality assumption met.

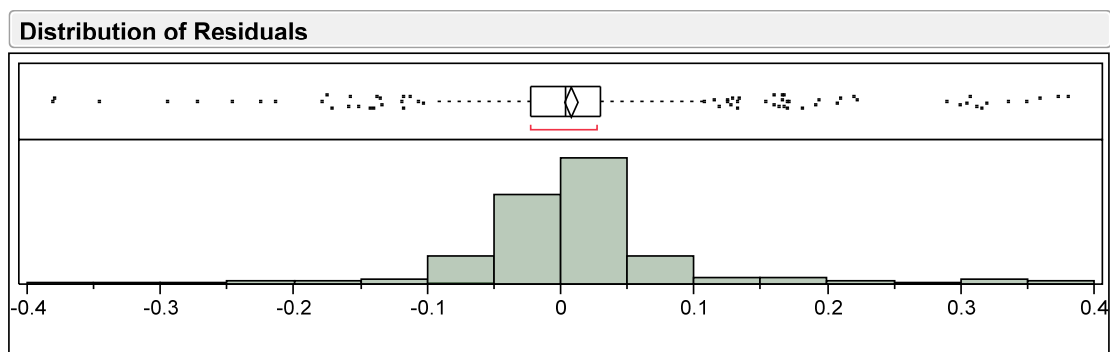


Figure 75. Parametric Model Distribution of Residuals

100.0%	maximum	0.3797
99.5%		0.3504
97.5%		0.1770
90.0%		0.0758
75.0%	quartile	0.0293
50.0%	median	0.0034
25.0%	quartile	-0.0228
10.0%		-0.0577
2.5%		-0.1292
0.5%		-0.3024
0.0%	minimum	-0.3809

Table 86. Parametric Model Distribution of Residuals Quartile Range

Mean	0.0073328
Standard Deviation	0.0737618
Standard Err Mean	0.002653
Upper 95% Mean	0.0125408
Lower 95% Mean	0.0021248
N	773

Table 87. Parametric Model Distribution of Residuals Moments

Given these statistics, 95% of the time the mean residual distribution between the model and the actual observation of a sample will be within $0.7\% \pm 0.5\%$ at a 95% significance level (DeVore, 2008).

- Variance $s^2 = \frac{\sum_{i=1}^n (y_i - \bar{y})^2}{n-1} \approx \frac{4.2002976}{772} \approx 0.0054408$
- Standard Deviation $s = \sqrt{s^2} = \sqrt{0.0054408} \approx 0.0737618$
- Standard error(SE) $= \frac{s}{\sqrt{n}} = \frac{0.0737618}{\sqrt{773}} \approx 0.002653$
- 95% Two-Tailed Student-T statistic with degrees of Freedom = 772 is approximately = 1.965
- Confidence Interval estimator:
 $\bar{y} \pm t_{n-1}^* SE(\bar{y}) = 1.965 * 0.002653 \approx 0.005213145$

i. Parametric Model Comparison to Sample Data

A direct comparison of the model and actual data is contained in Table 88. A visual inspection of the results shows that the model tends to bias towards better compression ratios based on the dir column has more - indicators. The “Dir” column is a direction of error indicator with a “-” meaning an error towards better compression and “+” meaning an error towards worse compression.

The higher count of “-” predictions is due to the equal weighting score function used for parametric regression, and that the EXI technique tended to deliver results at the mean or less. Generally, those cases that are above the mean tend to be way above the mean though in fewer numbers. The equal weighting score function on the EXI distribution artificially drives the model results down.

FileName	EXI Actual	Model Prediction	Dir	Residual	MD	NE	CE	NA	CA	NV	CV	CD
AllocationInstruction.xml	0.656	0.341	-	0.315	3	8	10	31	40	34	40	0.363
OneDisPacket.xml	0.500	0.294	-	0.207	4	13	13	31	44	31	44	0.159
teapot.x3d	0.257	0.400	+	0.144	6	9	9	8	10	10	10	0.996
wsdl1.xml	0.127	0.245	+	0.118	7	22	39	12	43	31	44	0.218
Snippet2-5.xml	0.364	0.280	-	0.084	6	14	17	10	19	19	19	0.153
ws-addressing-12.xml	0.386	0.331	-	0.055	5	11	11	0	0	6	6	0.226
w3cWebpage.xml	0.218	0.177	-	0.041	8	29	749	27	934	939	1584	0.447
libyan_arab_jamahiriya.svg	0.469	0.430	-	0.039	2	4	4	16	23	21	23	0.266
PortsWithBerthsEx1.xml	0.241	0.207	-	0.034	12	20	33	8	18	13	18	0.111
visitRequest.xml	0.173	0.154	+	0.019	5	51	87	1	2	50	57	0.071

Table 88. Parametric Model Comparison of Results–Score of Model

j. Parametric Model Conclusions

Overall, the parametric model delivered a good fit of the data based on the relatively high adjusted R-squared value, but cannot deliver a significantly small window in prediction ability as 15% of the variability is not accounted for. For a general impact study, this parametric model is well suited due to its simplicity and fair accuracy. For specific XML-language cases where high levels of accuracy are required, it may not provide the needed requirements of accuracy.

The model does indicate that it will generally be within 1% of the actual results, but as shown in Figure 75, the range of the distribution, though seldom, does go to $\pm 40\%$. However, reviewing quartile breakdown in Table 86, 80% of the sample errors are within 5% of actual. The occurrences of the extremes errors greater than 10% is only the upper and lower 2.5% of the range.

5. Non-Parametric Prediction Model

Non-parametric models are not considered accurate predictors, but they can work on any data regardless of its underlying distribution or lack of distribution. Non-parametric models reduce the require assumptions of parametric models which often leads them to being effective predictors with easy to understand models.

a. Non-Parametric Model Design Points

The non-parametric model technique chosen is a Classification and Regression Trees (CART). This type of model splits a data set into clusters based on if-then logical conditions, taking on a tree-like form called a Dendrogram, dissimilarity graph or tree (Hand, Manilla, & Smyth, 2001; Hastie, Tibshirani, & Friedman, 2009). Each branch and leaf of the tree consists of those observations that are the most in common with each other relative to all the other observations in the data set using a greedy approach algorithm. The greedy algorithm is roughly a RSS score function for each leaf of the tree, which is the mechanism that enables CART to overcome large variances and dissimilarities in data sets; splitting the data around multiple RSS that

maximizes the total R-squared value of the model. However, despite the fact that the model uses a greedy approach, it cannot be assured to converge to any specific optimum. That is, depending on the training data set, the tree that is grown could be different between runs.

Because CART is a non-parametric technique, eliminating the parametric model required assumptions. Therefore, data transformations (natural logarithms) are not needed or desired; the raw untransformed factor space and results data are used.

The fundamentals behind the CART technique are the splitting of the data set at predictor variable locations that have the lowest Residual Sum of Squares (RSS)

where $RSS = \sum_{i=1}^n (y_i - \hat{y}_1)^2$. This is called impurity reduction in that the variance of the predictability of the model is reduced, or otherwise described, the accuracy of prediction increases at each split of the data set; reduced variance equals better model performance (Hand, Manilla & Smyth, 2001; Hastie, Tibshirani, & Friedman, 2009). The measure of effect for a CART model is the R-squared term similar to the parametric model and is interpreted the same; the amount of variance explained by the model. The more splits, the more variance that is explained in the CART model, but the more splits comes at a cost of increased complexity of the model. The splitting of the data can continue up to N splits where N is the number of observations in the data set, which would leave one observation in each leaf. This would be a complex model and would not capture the underlying spirit of the data set, making it good for the training data, but extremely poor for testing data. A better stopping condition than N is something between N and 1 balanced by complexity. Often the approach used is when the R-squared delta between splits is low, below a user-defined epsilon. Note, that with each split (k) there will be k+1 leafs. The Depth and Breadth of the tree is dependent on the underlying data set, number of factors and variability in the data.

The positives for a CART model:

- Easy to explain and implement.
- Graphical output.
- Including categorical predictors is easy.
- Transformations or scores for ordinal categorical variables make no difference in the model results.
- Robust to outliers.
- Interactions are included automatically.
- Irrelevant predictor variables don't cause trouble.

The negatives for a CART model

- Difficult to interpret.
- Not stable between training sets.
- Poor predictive power.
- Trees do not capture simple additive structure.
- Trees do not give smooth estimates.

b. Non-Parametric Model Developed

The CART model generated using the test-corpus consists of 12 splits of the data set. The R-squared measurement of effectiveness of the model in explaining the data set is 0.745, Figure 76. The 12 splits stopping condition is based on the low delta R-square difference in subsequent splits of the data set. This decreasing delta can be seen in Figure 76 by noting the decreasing slope between splits, shown in the highlighted area.

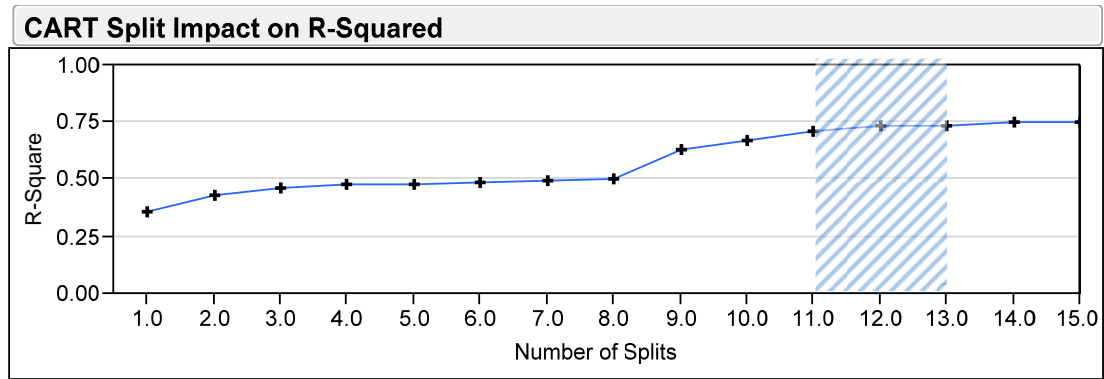


Figure 76. Non-Parametric Model Split History

The resulting tree model is contained in Figure 77 with statistics at each branch and leaf. Figure 78 contains the same tree, but without the leave node statistics making for a slightly easier view.

The tree is traversed from the root to a leaf node based on the if-then conditions at each branch that correspond to the XML document being modeled. Overall, the tree is a series of AND conditions that describe the document until a leaf node is reached. The predicted value for the document is the mean value of the terminal leaf.

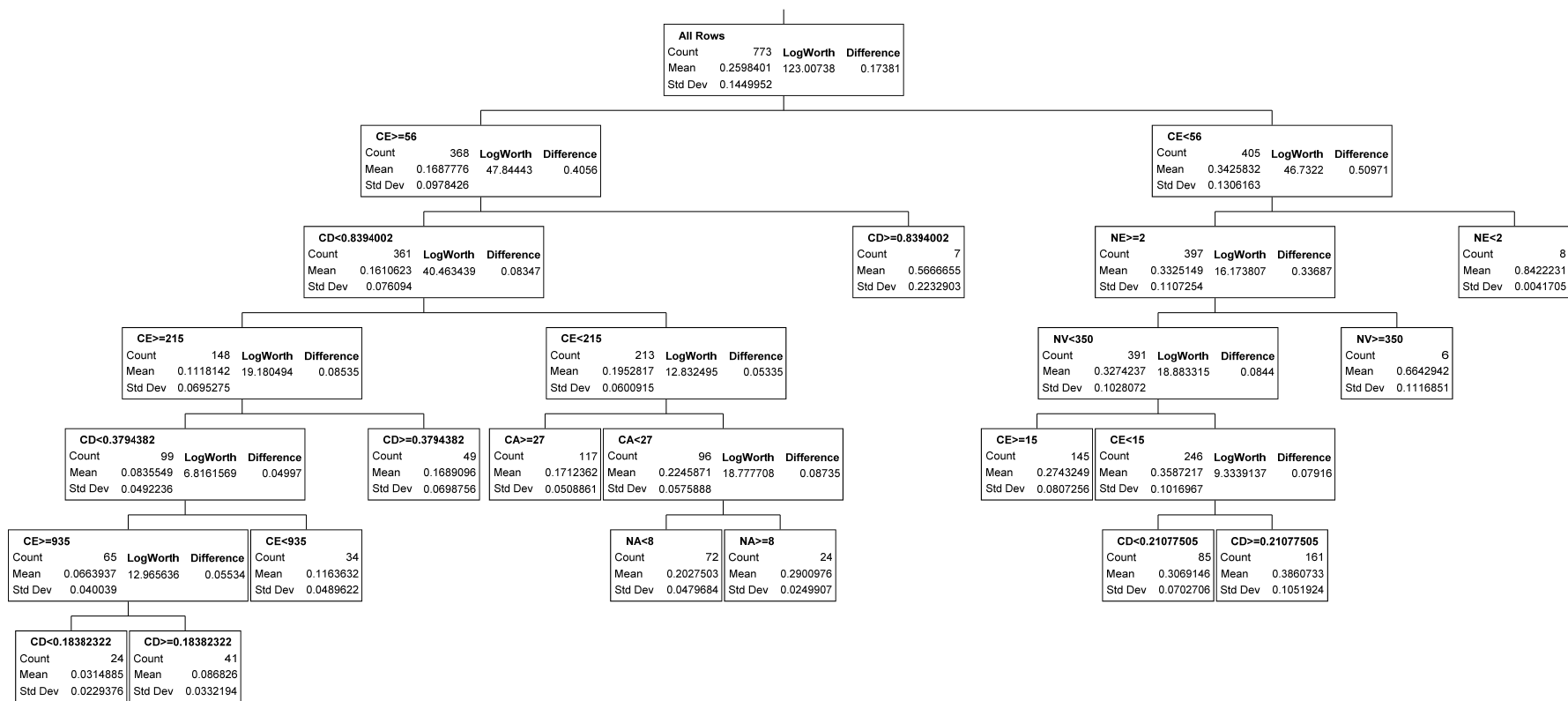


Figure 77. Non-Parametric Detailed CART Tree

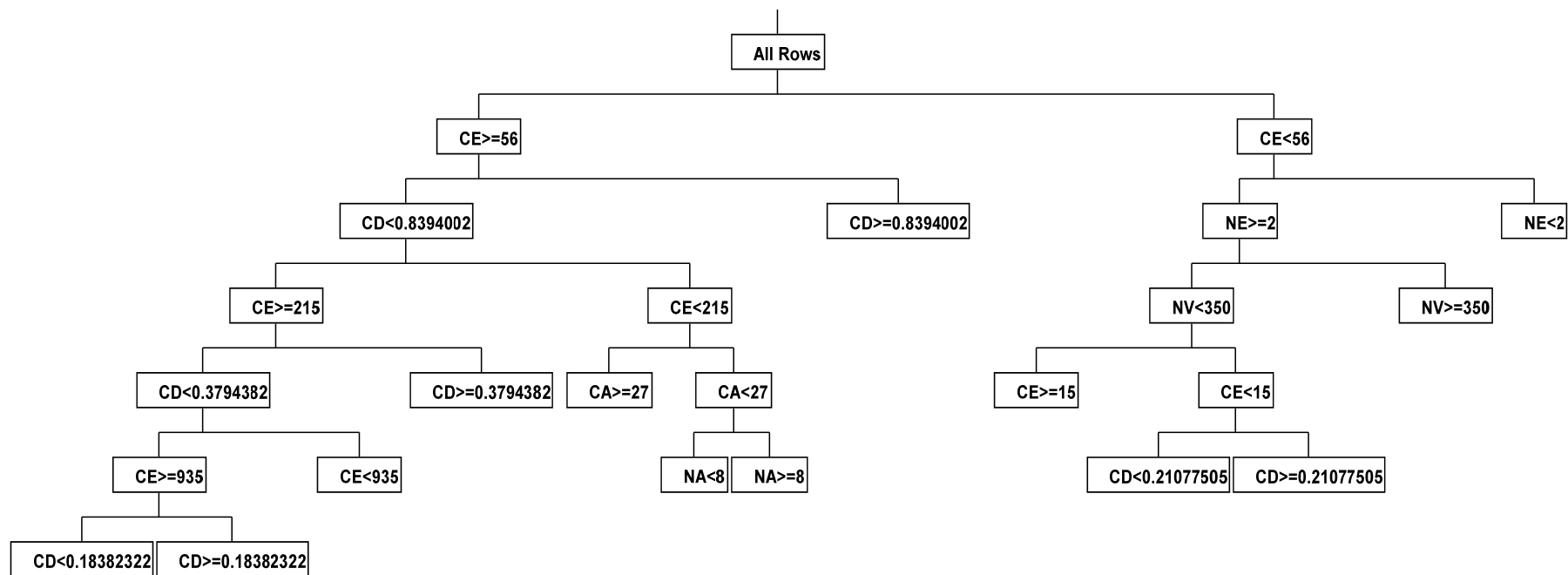


Figure 78. Non-Parametric Simple Branching CART Tree

c. Non-Parametric Model Results Interpretation

To explain the CART results, Table 89 is provided with all paths from root to leaf (left to right model leaf breadth) with predicted values at each leaf.

- An indication to the strength of a leaf node's prediction is the count of observations contained within the node. The higher the count the more precise the prediction will be.
- The total number of leaves is 13, which is the 12 splits + 1 as defined in the introduction of the CART technique.
- Key to point out about CART is that all XML documents with the same factor space settings will have the same prediction (categorized), which is why non-parametric models are often inaccurate predictions, but appealing when used with noisy and high variance data.

Leaf Number (L -> R)	Root Path to Leaf				Prediction (Mean)	Count
1	CE >= 56	&	CD < 0.8394002	&	0.03148845	24
	CE >= 215	&	CD < 0.3794382	&		
	CE>=935	&	CD < 0.18382322			
2	CE >= 56	&	CD < 0.8394002	&	0.08682599	41
	CE >= 215	&	CD < 0.3794382	&		
	CE >= 935	&	CD >= 0.18382322			
3	CE >= 56	&	CD < 0.8394002	&	0.11636318	34
	CE >= 215	&	CD < 0.3794382	& CE < 935		
4	CE >= 56	&	CD < 0.8394002	&	0.16890961	49
	CE >= 215	&	CD >= 0.3794382			
5	CE >= 56	&	CD < 0.8394002	&	0.17123616	117
	CE < 215	&	CA >= 27			
6	CE >= 56	&	CD < 0.8394002	&	0.20275031	72
	CE < 215	&	CA < 27	& NA < 8		
7	CE>=56	&	CD<0.8394002	&	0.29009756	24
	CE<215	&	CA<27	& NA >= 8		
8	CE >= 56	&	CD >= 0.8394002		0.56666547	7
9	CE < 56	&	NE >= 2	&	0.2743249	145
	NV < 350	&	CE >= 15			
10	CE < 56	&	NE >= 2	&	0.30691457	85
	NV < 350	&	CE < 15	&		
	CD < 0.21077505					
11	CE < 56	&	NE >= 2	&	0.38607334	161
	NV < 350	&	CE < 15	&		
	CD >= 0.21077505					
12	CE < 56	&	NE >= 2	& NV >= 350	0.66429417	6
13	CE < 56	&	NE < 2		0.84222307	8

Table 89. CART Leaf Summary

For example, using the same test as the parametric model, if a sample XML document had the following characteristics the model result is 0.1163:

- (MD) Depth = 6
- (NE) Number of Unique Elements = 127
- (CE) Count of Elements = 229
- (NA) Number f Unique Attributes = 2
- (CA) Count of Attributes = 4
- (NV) Number of Unique Values = 78
- (CV) Count of Values = 152
- (CD) Content Density = 0.04900332

d. Non-parametric Model Analysis of Fit and Feasibility

Inspecting the CART model reveals that it has similar results as the parametric model previously presented.

Figure 79 shows the impact of each factor on the CART model. Similar to the parametric model (Figure 72), the depth of the document (MD) has no impact on the model, and the count of elements (CE) has the largest impact on the model. These are followed by number of unique elements (NE) and content density (CD) both similar in result to the parametric model.

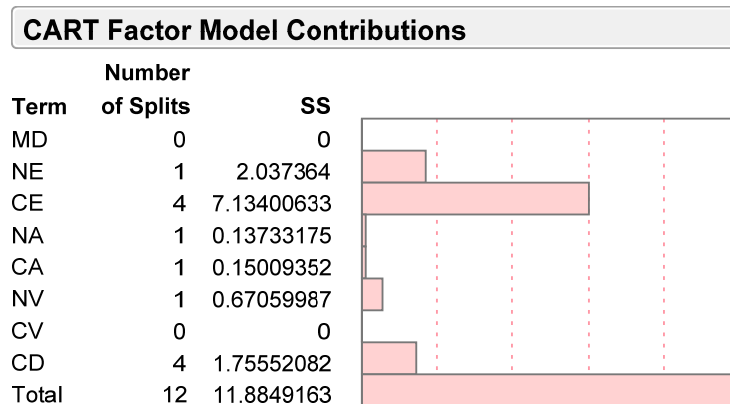


Figure 79. Non-Parametric CART Factor Effect on Model

Unlike a parametric, the only real measure of feasibility other than the R-squared value is the residuals, or the error of the model. Figure 80 is a plot of the residual values distribution, that is actual – predicted, and both Table 90 and Table 91 are the statistics of the distribution: a mean of 0 and standard deviation of 7% (standard error of 0.2%) of the predicted value to the actual result. Very similar to the parametric model back in Figure 75, the shape has a normal looking distribution.

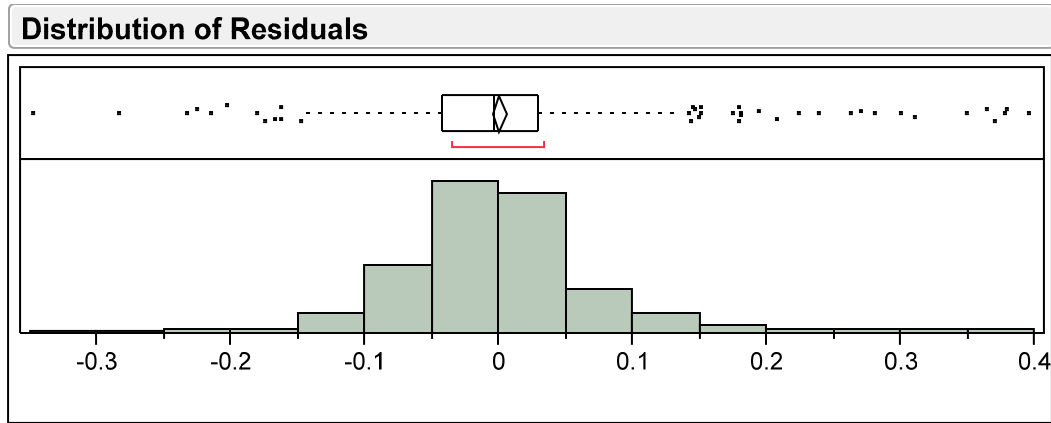


Figure 80. Non-Parametric CART Model Distribution of Residuals

100.0%	maximum	0.3960
99.5%		0.3710
97.5%		0.1790
90.0%		0.0774
75.0%	quartile	0.0291
50.0%	median	-0.0030
25.0%	quartile	-0.0415
10.0%		-0.0696
2.5%		-0.1300
0.5%		-0.2265
0.0%	minimum	-0.3484

Table 90. Non-Parametric CART Model Distribution of Residuals Quartile Range

Mean	0
Standard Deviation	0.0750243
Standard Err Mean	0.0026984
Upper 95% Mean	0.0052971
Lower 95% Mean	-0.005297
N	773

Table 91. Non-Parametric CART Model Distribution of Residuals Moments

Given these statistics, 95% of the time the mean residual distribution between the model and the actual observation will be $0\% \pm 0.5\%$ (DeVore, 2008).

- Variance $s^2 = \frac{\sum_{i=1}^n (y_i - \bar{y})^2}{n-1} \approx \frac{4.345356}{772} \approx 0.0056287$
- Standard Deviation $s = \sqrt{s^2} = \sqrt{0.0056287} \approx 0.0750243$
- Standard error(SE) $= \frac{s}{\sqrt{n}} = \frac{0.0750243}{\sqrt{773}} \approx 0.0026984$
- 95% Two-Tailed Student-T statistic with Degrees of Freedom (df) = 772 is approximately = 1.965
- Confidence Interval
estimator: $\bar{y} \pm t_{n-1}^* SE(\bar{y}) = 1.965 * 0.0026984 \approx 0.005302356$

e. Non-Parametric Model Comparison to Sample Data

A direct comparison of the model and actual data using the same samples as the parametric model (Table 88) is listed in Table 92. The CART seems to be more centered in its error with equal “+” and “-” compared to the parametric regression model.

FileName	EXI Actual	Model Prediction	Dir	Residual	MD	NE	CE	NA	CA	NV	CV	CD
AllocationInstruction.xml	0.656	0.386	-	0.270	3	8	10	31	40	34	40	0.363
OneDisPacket.xml	0.500	0.307	-	0.194	4	13	13	31	44	31	44	0.159
teapot.x3d	0.257	0.386	+	0.130	6	9	9	8	10	10	10	0.996
wsdl1.xml	0.127	0.274	+	0.147	7	22	39	12	43	31	44	0.218
Snippet2-5.xml	0.364	0.274	-	0.090	6	14	17	10	19	19	19	0.153
ws-addressing-12.xml	0.386	0.386	0	0.000	5	11	11	0	0	6	6	0.226
w3cWebpage.xml	0.218	0.168	-	0.050	8	29	749	27	934	939	1584	0.447
libyan_arab_jamahiriya.svg	0.469	0.386	-	0.083	2	4	4	16	23	21	23	0.266
PortsWithBerthsEx1.xml	0.241	0.274	+	0.033	12	20	33	8	18	13	18	0.111
visitRequest.xml	0.173	0.203	+	0.030	5	51	87	1	2	50	57	0.071

Table 92. Non-Parametric Model Comparison of Results

f. Non-Parametric Model Conclusions

Overall, the non-parametric model delivers a good fit of the data based on the R-squared value. Similar to the parametric model, this CART model is suited at best for exploratory analysis of EXI. However, given the simpler implementation and explanation of the CART model, CART may be better suited for experimental analysis than the parametric model due purely to simplicity.

Since CART is a clustering prediction model, the conglomerate predictions will not lead to long-term accurate results between testing sets. The only way to begin to overcome this is through massive N, an N value that can genuinely capture every possible combination of XML factors and variance. CART works best on larger sets of data, and the more variance, as in any model, the more data observations that are needed to increase accuracy. However, the model does indicate it will generally be within 0.5% of the actual results, but as shown back in Figure 80, the range of the distribution, though seldom, does go to $\pm 40\%$. However, reviewing quartile breakdown in Table 90, 80% of the sample errors are within 7% of the actual value. The occurrences of the extremes errors greater than 10% is only the upper and lower 2.5% of the range.

6. Conclusions Regarding EXI Prediction Models

a. Significance between the General Models

Both models generated statistically similar results but with structurally different forms. The parametric model averages a 0.7% ($\pm 0.5\%$) difference between actual and predicted results and the non-parametric model 0% ($\pm 0.5\%$). Although the non-parametric model has a lower residual average, its inner quartile range is larger than the parametric model implying more variation in the model's prediction ability. A larger variation in prediction ability from a parametric model is as expected given the classification like process of the non-parametric CART model. However, statistically, with 95% confidence, the two models are not different. Figure 81 is an analysis of the two models in their prediction accuracy, highlighting the fact that neither model is

statistically different from the other. The green 95% confidence interval diamonds overlap as do the Tukey-Kramer multiple comparisons, both of which demonstrate that neither model is statistically better. Note that the y-axis is scaled to make the confidence intervals more obvious. Prior to scaling, due to the narrow confidence intervals of the models relative to the range of the residuals, it was difficult to make out the details of the graphic. However, this scaling does not detract from any of the impact of the ANOVA as the diamonds and Tukey-Kramer circles are the important aspects of the image and not the full range of values. Refer back to Table 86 for the full data range of the parametric model and Table 90 for the full data range of the non-parametric as needed.

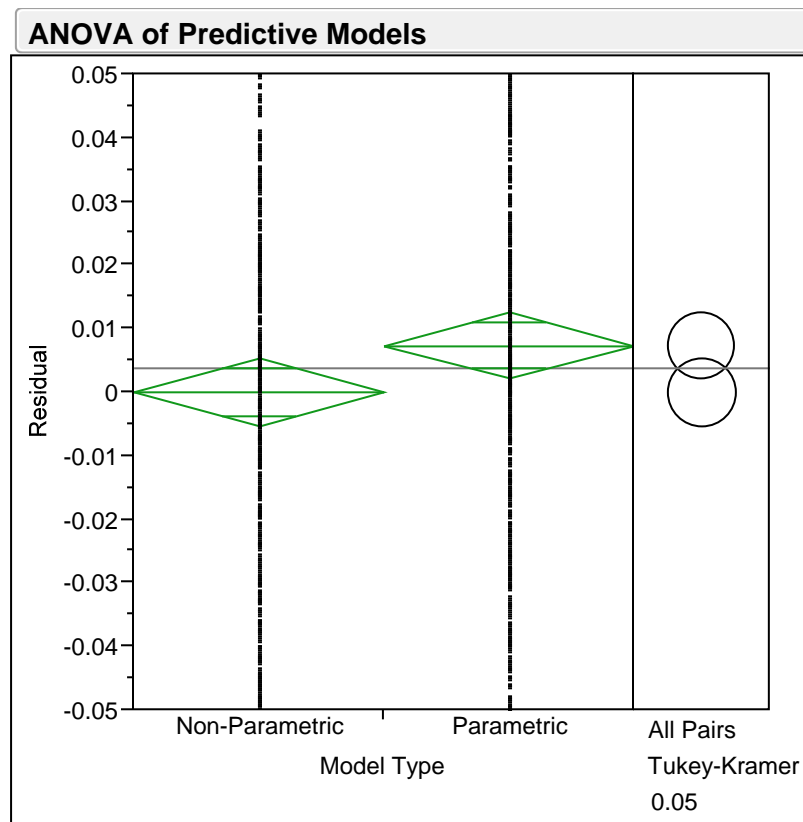


Figure 81. Analysis of Variance (ANOVA) Comparison of Predictive Models

b. Domain-Specific Models

A domain-specific model does indicate the possibility of providing better predictive abilities, but not at a significant level. To demonstrate the domain-specific model concept, a X3D domain model was created using the 197 X3D example scenes from the X3D for Web Authors book chapters (Brutzman, 2007). Though not presented in same verbose manner as the other models, the parametric X3D model produced had an approximate R-squared value of 0.88 and the non-parametric X3D model produced an approximate R-squared value of 0.83. The X3D specific parametric model's predictive ability is only slightly better than the model produced with the general XML test-corpus, but the non-parametric results are noticeably improved capturing almost 10% more of the data's variance. The ANOVA in Figure 82 highlights the significance between the general and domain-specific models and Table 93 list the distribution statistics of the four models.

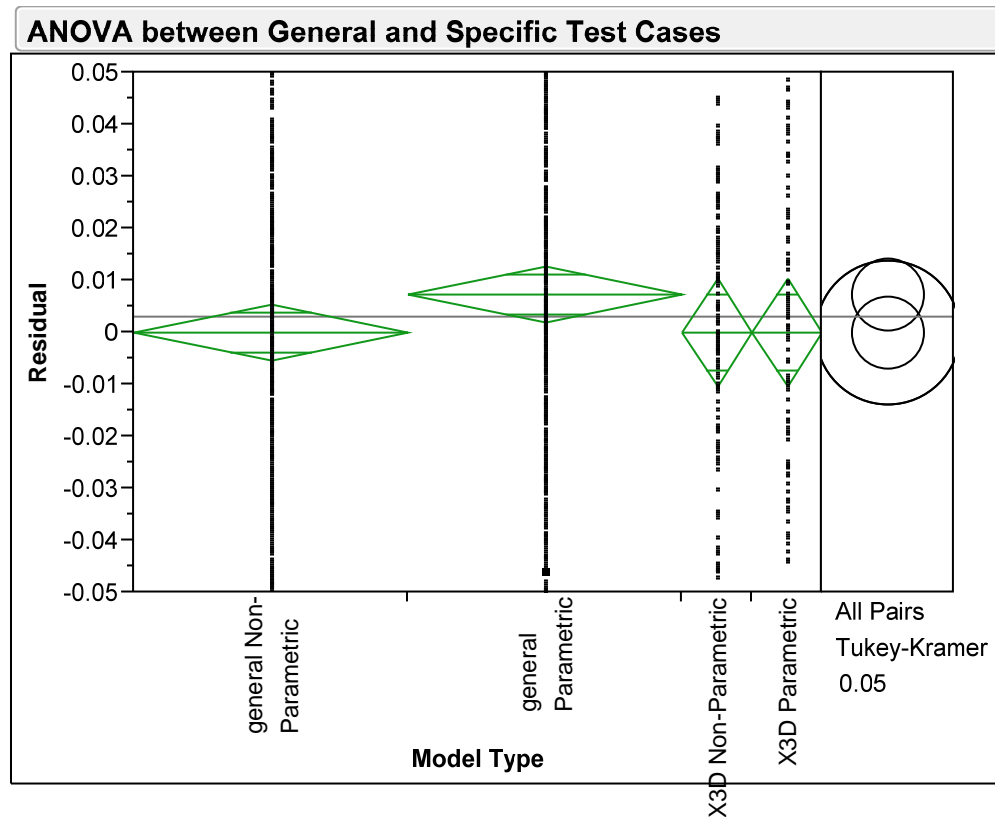


Figure 82. Analysis of Variance Between Parametric and Non-Parametric Models for General and Domain-Specific XML Cases

Model Type	N	Mean	Standard Error	Lower 95%	Upper 95%
CART_GEN	773	0	0.0026984	-0.005297	0.005297
CART_X3D	197	0	0.0029447	-0.005807	0.005807
LSE_GEN	773	0.007	0.002653	0.0021248	0.0125408
LSE_X3D	197	0	0.007496	-0.01479	0.01479

Table 93. Domain-Specific X3D Models Comparison to General XML Models Statistics

The results of the domain-specific models are slightly better R-squared values with reasonably consistent variance between the general and domain-specific models. Although not statistically significant, the increased R-squared will likely enable more accurate predictions for specific document domains as would be expected when sampling from a narrower more-consistent range of XML documents.

c. EXI Models in General

Parametric models are generally accurate, but they require several assumptions about the data that might not be possible for all XML domains, though in general XML tends to meet the assumption requirements. Non-parametric models remove the data assumption requirements of parametric models, but at the cost of reduced accuracy. Parametric regression and non-parametric trees are the preferred modeling methods to employ when lacking other prior knowledge about the data that would otherwise indicate an alternative model. Other types of models might possibly deliver better prediction abilities for EXI, but they tend to come at the cost of both implementation complexity and lack of simple (or even consistent) explanation. However, regardless of model type, the high variance within XML documents will negatively affect the accuracy of any prediction model. Which type of model is used to predict EXI performance depends on the underlying domain characteristics of its data, the implementer's familiarity with the technique, and the intended research questions that need to be answered.

Well-structured XML domains that contain large percentages of repetitive data will achieve good or better predictive abilities using either the parametric or non-parametric model techniques. Domains that do not have a predictable structure will likely require a non-parametric model to achieve any degree of predictive certainty in order to mitigate the variance.

Highly accurate models of EXI's performance on XML in general is not obtainable, and a domain-specific model is potentially achievable, but not likely. Ultimately, a precise model that predicts to within 0.5% of the actual at a 99% level is desired, but due to the innate variance of XML, is not achievable in general.

EXI predictive models are best suited for exploratory analysis of EXI's effect for potential EXI adopters. The models satisfactorily answer whether or not EXI will deliver statistically better results compared to a domain's existing practices.

H. EXI IMPLEMENTATIONS AND TOOLS

1. Available EXI Implementations

Several EXI implementations are being developed in parallel at different levels of completeness:

- The initial format authors, Agile Delta, have a commercial implementation in both Java and C++ (Agile Delta, 2009).
- Siemens Corporation has developed an open source EXI implementation in Java, licensed as GPL (a viral license). Their implementation can be downloaded from the Siemens' SourceForge site [<http://exificient.sourceforge.net/>] (Siemens, n.d.).
- The Naval Postgraduate School (NPS) has produced another open-source implementation under the Apache open-source license, reported in detail within this thesis [openexi.sourceforge.net]. The ultimate goal of the NPS implementation is to integrate it with other contributors into Apache as an Apache project.

2. NPS EXI Comparison Tool

The Naval Postgraduate School in its effort to evaluate the effectiveness of the EXI solution for DoD specific applicability created a tool to compare EXI compactness to other common DoD compression formats. Figure 83 is a screen shot of the interface and sample results of a run. It automatically produces comparison of no operation, GZip, Zip, EXI schemaless, and when available schema-informed compression of an input XML file. The results of each technique are stored with original name with the technique extension in the /data subdirectory. This comparison tool, encoded in Java, is freely available from the NPS MOVES Institute Web page <http://www.movesinstitute.org/exi/EXI.html>.

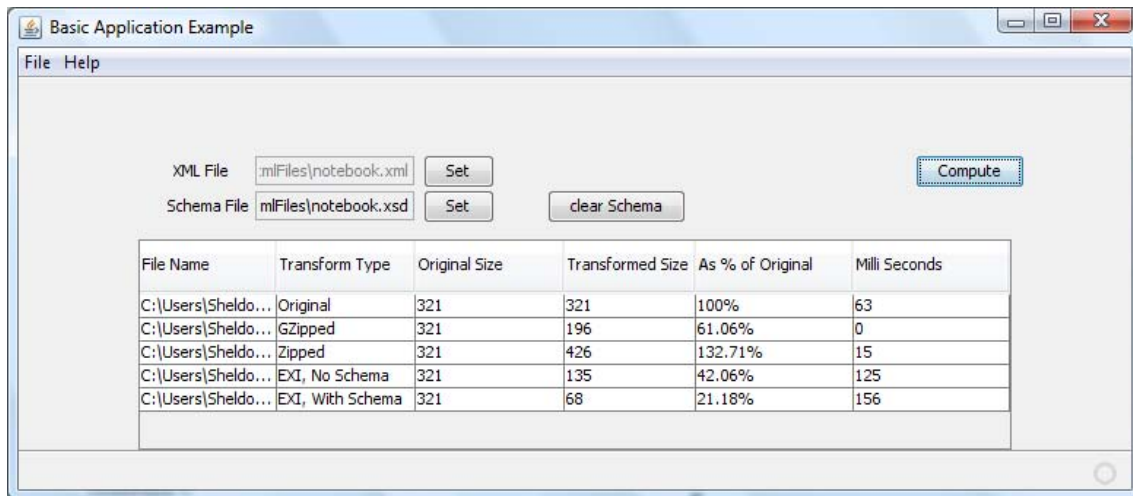


Figure 83. Technique Comparison Tool

3. NPS Options Tool

In addition to the compression comparison, a Graphic User Interface (GUI) that exercises the EXI encoding options was build using the Siemens codebase engine Figure 84. This tool allows the exercise of the available EXI options for both schema-informed and schemaless EXI encodings. This options exercising tool, encoded in Java, is freely available from the NPS MOVES Institute Web page <http://www.movesinstitute.org/exi/EXI.html>.

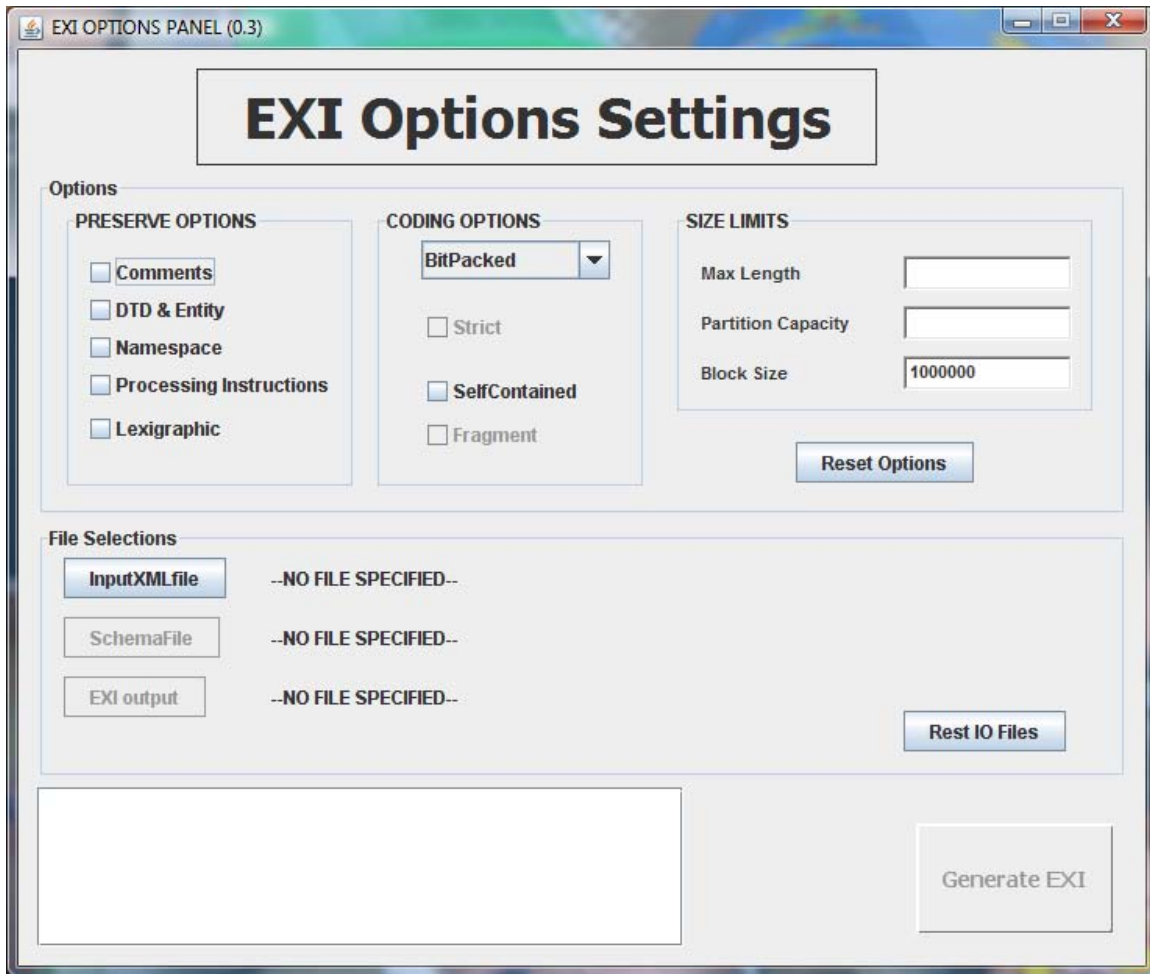


Figure 84. EXI Options Tool

I. CHAPTER CONCLUSION

EXI schema-informed compression delivers superior results statistically compared to all compression techniques used within DoD. EXI schemaless, when compared to the other techniques is always at least equal to the next best compression technique. With EXI, the DoD can expect a doubling of bandwidth potential based on the assumption all network communications are compressed pre-transmission with the efficiency of the GZip technique. In cases where there is no pre-transmission compression (raw file transmissions), EXI has an expected 5x increase in bandwidth potential. The amount of native variance within XML documents makes predicting EXI performance difficult to accomplish with any high degree of accuracy.

J. CHAPTER SUMMARY

The chapter starts by defining a sampling of DoD relevant XML cases and demonstrating the effect EXI has on these files. Recommended configurations of EXI are then presented with a focus towards optimal bandwidth savings. A test-corpus of 773 XML documents is defined in terms of a variety of XML families and applications. Using the test-corpus, the statistical effect of EXI is demonstrated with an Analysis-of-Variance (ANOVA). The test-corpus is then used to build both parametric and non-parametric predictive statistical models. The chapter concluded with a listing of available EXI implementations and tools, and where they can be downloaded, many freely available with source-code.

X. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

Based on the research conducted for this thesis, the DoD will directly benefit from an EXI solution within its networks in support of the Network-Centric data sharing strategy of XMLizing the GIG. This thesis has shown that XML is an extremely powerful and widely embraced data-representation format that delivers system-to-system interoperability, but does so at the cost of being verbose and often complex to process. These costs of XML can prevent network edge devices from performing real-time network operations due to limited bandwidth and processing capabilities constrained by small CPUs and limited memory capacity. However, through the EXI technique's alternative XML format, the verbosity and processing cost of XML are shown to be significantly reduced, enabling real-time network operations by edge devices.

Ultimately, if EXI is widely implemented, the DoD will be able to deploy XML-based network traffic further, specifically to the individual sailor or soldier on handheld or other small mobile and wireless network edge devices. Through this expanded network penetration, the DoD will realize a more informed and tactically aware force able to better project power and defend our country.

1. The Technology Development and Adoption Litmus Test

EXI passes the "More-Faster-Better" litmus test for modern technology development and adoption:

- **MORE**—EXI delivers more potential bandwidth capacity through compression levels that are statistically superior to all other techniques. Specifically for the DoD XML domain EXI has shown an expected doubling of bandwidth potential; enabling more (deeper) network penetration with all the benefits of XML.

- **FASTER**—The binary representation of EXI is effectively equal to or better than traditional IEEE formats, and is faster than the native XML format. EXI enables an efficient XML data representation that delivers faster processing and faster data exchanges with all the benefits of XML.
- **BETTER**—One of the most important aspects of EXI is that it is able to directly integrate into the existing XML stack and network architectures, and XML is how data is represented across the IT gamut. EXI does not require the reengineering of data or physical network architecture, it simply “extends” what already exists and provides backwards as well as future compatibility. EXI is better because it gives more with what you already have and does it faster, transparently.

2. Research Questions Answered

a. Can the Department of Defense (DoD) Keep Up with Enterprise America’s Constantly Connected Internet Applications Philosophy?

EXI is the answer to the problem. As the business world implements “Cloud” computing philosophies for distributed data, they are relying on XML as the fundamental bases for data formats. A quick review of many of these services reveals the underlying architecture is Web services wrapped in XML. Because our low-bandwidth units must access these services, but cannot properly support the bandwidth demands, either a physical increase in bandwidth must be provided to every unit, or a technology that compresses data well enough that a bandwidth increase is realized is required to ensure DoD keeps pace with the business world.

Physical bandwidth requires both new satellites and antenna systems mounted on every ship and vehicle within DoD. Physical bandwidth is the optimal solution, but is not financially or physically feasible. EXI delivers the next best approach through XML compression that enables a bandwidth increase through file size reductions, and is backed by the realization of the increasing usage of XML in the IT world.

b. Can DoD Data, using the Extensible Markup Language (XML), be Efficiently Compressed to a Level that Makes Porting to and from Low-bandwidth Military Units Feasible?

By implementation EXI at the transmitting and receiving stations, this goal becomes feasible. The EXI technique has shown that extremely large XML files can be compressed to fractions of a percentage of their original size. For example the medical test case Medical_G4Data3.heprep is a 34MB XML file, but when compressed by EXI ended up as a 1.2MB file, or 2% of the original. Medical, M&S, personnel and nearly all other data transactions within DoD are XML-based and, are large file. For deployed units such as navy ships operating on essentially legacy dialup connectivity, the only options to update files is to wait until the ship returns to land, time-late. With EXI, the real-time transfer of numerous larger XML files becomes a realistic and doable expectation even under the low-bandwidth restrictions of deployed units.

As XML is implemented in more and more domains by DoD directive, the file size problem of XML will be exacerbated without EXI. EXI enables the efficient compression XML needs to support the XML demand of DoD in the low-bandwidth environments.

c. What are the Risks to the DoD Infrastructure if they Do Not Develop Methods to Push Data Further Down the Echelon Chain of Command?

The risks are enormous if DoD does not continue to push information further down the networks, and ultimately down to the individual sailor and soldier. Our counterterrorism and other asymmetric warfare actions are extremely time critical, demanding real to near real-time information in order to combat the asymmetric threat. The source of the risk to real-time data sharing is XML itself due to its size and computational processing complexity; posing bounds on the network penetration depth. Using EXI removes these bounds and enables information to reach the farthest deployment devices.

Information will be pushed to the edge with or without XML, but without XML the risks are the loss of interoperability, and the creation of next generation of stove pipes; exactly what the ASD mandated XML usage policy is working to prevent. Without information at the edge, it increases the potential for our enemies to become a more connected force in battle, achieving cyber dominance over our American forces. This risk is increasing as the business world, the providers of DoD IT systems, are moving towards a constantly connected IT structure that demands constant data flow, with XML as the backbone.

EXI is the solution to provide US forces the ability stay constantly connected at the network edge.

d. Is XML an Effective Tool for Commands that are Under Extremely Low-bandwidth Constraints?

XML is the proven method to achieve interoperability and is the driving force behind the Network-Centric and Force-Net visions of DoD. XML is a well understood and supported data-structure architecture that is flexible (extensible) to meet both the needs of today as well as tomorrow. XML is in line with the today's IT business models and practices; XML is how IT is done. Without XML, the DoD will be regressing backwards to a time with multiple incompatible data standards. EXI enables XML to continue to be fully and seamlessly implemented across the network.

B. RECOMMENDATIONS FOR FUTURE WORK

1. Full EXI Specification of OPENER-EXI

The OPENER-EXI codebase is only a partial specification implementation; complete the remaining EXI specification section such as compression and schema-informed processing. As of this thesis, OPENER-EXI and an industry partner are establishing a collaborative EXI implementation under the Apache license with intent to develop the mature code as an incubator project in the Apache Software Foundation.

2. Develop a Micro Version of EXI

Implement the OPENER-EXI code set within the Java 2 Micro Edition (J2ME) and install it on a handheld device or cell phone. J2ME is a subset of the larger Java language designed for efficient operations on smaller and low-power devices.

It is likely that the transition from the Java 2 Standard Edition (J2SE) current implementation coded set into the J2ME will have some library conflicts. A majority of the OPENER-EXI code set uses basic datatypes (int and Strings), so the number of data-structures should be minimal. One of the key data-structures, HashMap, used in the OPENER-EXI implementation is also in the J2ME so the transition should require minimal effort, but will have some nuances that will have to be overcome. A quick check test of the expected complexity that will be involved in such as transition can be estimated by compiling the OPENER-EXI code as is with the compiler -source flag set appropriately for the J2ME package. Based on the error messages received should lead to an estimate of effort required.

3. Create an Example of the Motivation Scenario

Build an example test of the proposed scenario presented in Introduction Chapter of this thesis.

- 1) Build the scenario using a simulation toolkit, such as SIMKIT, a Discrete Event Simulation (DES) toolkit, as the scenario simulator engine.
- 2) Host the simulator engine on a NPS server.
- 3) Install a micro version of the EXI specification on a handheld device.
- 4) Wirelessly communicate scenario parameters in EXI format from the handheld to the server hosting the simulation engine.
- 5) Execute the scenario an appropriate number of times and wirelessly transmit the results of the simulation in EXI format to the handheld device.
- 6) Once the 2-way communications and simulation engine are working, add a 3D visualization of the simulation using X3D (a XML language) as the visualization tool.

The point of this work will be the analysis of the round-trip performance as well as the ability to present a scenario in the EXI format on a handheld device. The ultimate success of this activity would be a rapid and fully 3D representation of the simulation on the handheld devices to include real-time updates and simulation play out, that is, demonstration and proof of concept.

4. DoD Integration Package

Assuming EXI is to be integrated into DoD, a full security accreditation package of EXI must be documented and submitted to appropriate Programs of Record. Build a preliminary security accreditation package for EXI accreditation and publish to a DoD Program of Record. A possible starting direction that may have interest in EXI is PMW 160, the afloat networks executive: SPAWAR San Diego, CA.

5. Demonstrate the EXI Processing

Create a visual representation of EXI's grammar learning processes within a Finite State Machine (FSM) visualization toolkit. A FSM is an approximate representation of the Chomsky process employed within EXI and there are a number of FSM toolkits that can represent FSM processes visually. This result will be helpful for debugging analysis and educational purposes.

6. Efficient EXI Fragments in Line with XML

A possible gap in the EXI specification is the need for a formal definition on how to embed an EXI fragment with an XML document so that the same XML document can then be digitally signed and or encrypted.

Some use cases need to compress only part of an XML document and retain the results (an EXI fragment) in context within a new XML document. For these cases it would also be likely that a digital signature and/or encryption of the new XML document might follow. Both XML-based digital signature and encryption require a fully valid XML input document; an EXI file is not signable nor encryptable because an EXI fragment is not a valid XML document.

A DoD example case of such a need is the Plan of the Day (POD) example that follows. The contents of the SequenceOfEvents sub elements of the PlanOfDay element are confidential, but the dayOfWeek and Title elements are not, able to be left in the clear. The goals of this exercise are to EXI the contents of the SequenceOfEvents elements and retain them as an EXI fragment within a new XML POD document for follow on digital signature and encryption.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlanOfDay>
  <dayOfWeek date="Monday 18 October 2009"/>
  <Title>Some Title for the day's Events</Title>
  <SequenceOfEvents>
    <Event>The first event of the day</Event>
    <Event>The second event of the day</Event>
    <Event>The third event of the day</Event>
  </SequenceOfEvents>
</PlanOfDay>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<PlanOfDay>
  <dayOfWeek date="Monday 18 October 2009"/>
  <Title>Some Tile for the day's Events</Title>
  <SequenceOfEvents>
    [EXI Fragment of SequenceOfEvents]
  </SequenceOfEvents>
</PlanOfDay>
```

Once a XML document is compressed to EXI it is no longer effectively compressible because the EXI fragment will have few if any redundant bytes; no compression algorithm will deliver good results. Additionally, EXI fragment will have none printable characters from the EXI event codes and associated table indexes (byte values less than 33, ASCII's first printable character value). These non-printing characters cannot be captured within an XML string since they are non-printing, and as such, portions of an in line EXI fragment will be lost.

Some potential solutions starting points for consideration:

- A base64 formatting such as SOAP Message Transmission Optimization Mechanism (MTOM) (W3C, 2005).
- XML-binary Optimized Packaging (XOP) (W3C, 2005).
- Also refer to Jeff Williams Thesis (2009) for details about XML encryption and signature.

7. Department of the Navy (DON) Needs to Join the W3C

The World Wide Web Consortium (W3C) mission is to “Lead the Web to its full potential.” Numerous technical activities of great importance occur as part of working group activity at the W3C. Since Web technologies are the heart of the Navy and DoD’s Global Information Grid (GIG) strategies, renewed membership will pay significant benefits and avoid major lost benefits. Of note is that this thesis would not have been possible without NPS participation in the XML Binary Characterization (XBC) and EXI working groups. These efforts were not sponsored by the Navy, but rather by NPS representation of Web3D Consortium interest in W3C. Similar benefits can become available to other Navy laboratories, schools and technical programs by rejoining W3C.

LIST OF REFERENCES

- 7Zip. (2009). Welcome to the 7-Zip home! Retrieved September 10, 2009, from <http://www.7-zip.org/>
- Abbassi, B., Snyder, S., & Stoner, D. (2003). Integration of SQL with XML syntax in relational database systems. *ACMSE'03 ACM Southeast Conference*, 41, 155–160.
- Adobe Systems, Inc. (2003, August 11). Position on the binary interchange of XML infosets. Retrieved September 17, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/28-adobe.pdf>
- Advanced Technologies Group, NDS. (2003). The use of binary representations of XML information sets in digital broadcasting systems. Retrieved November 3, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/06-NDS-Position-Paper.pdf>
- Agile Delta. (2003). Theory, benefits and requirements for efficient encoding of XML documents. Retrieved October 29, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/30-agiledelta-Efficient-updated.html>
- Agile Delta. (2009, December 10). Lightning-fast delivery of XML to more devices in more locations. Retrieved December 22, 2009, from: http://www.agiledelta.com/product_efx.html
- Arnold, Adrian D. XML tactical chat (XTC): Extensible messaging and presence protocol for command and control applications. Master's thesis, Department of Computer Science, Naval Postgraduate School, Monterey, California, September 2006.
- Apache (2004, January). The Apache Software Foundation: Apache license, version 2.0. Retrieved September 15, 2009, from <http://www.apache.org/licenses/LICENSE-2.0.html>
- Apache. (n.d.). A guide to proposal creation. The Apache Software Foundation: Retrieved September 15, 2009, from <http://incubator.apache.org/guides/proposal.html>

- Apache. (n.d.). The Apache Software Foundation: Incubation policy. Retrieved September 15, 2009, from http://incubator.apache.org/incubation/Incubation_Policy.html
- Apache. (n.d.). The Apache Software Foundation: Incubation guides. Retrieved September 15, 2009, from <http://incubator.apache.org/guides/>
- Apache. (n.d.). The Apache Software Foundation: How the ASF works. Retrieved September 15, 2009, from <http://www.apache.org/foundation/how-it-works.html#what>
- ASD NII. (2006, April 12). Guidance for implementing net-centric data sharing (DoD 8320.02-G). Washington, DC: DoD. Retrieved from DTIC online: Information for the Defense Community Web site: <http://www.dtic.mil/whs/directives/corres/pdf/832002g.pdf>
- ASD. (2007, April 18). Cost and software data reporting (CSDR) manual (DoD 5000.04-M-1). Retrieved September 10, 2009, from United States Department of Defense Web site: <http://www.js.pentagon.mil/whs/directives/corres/pdf/500004m1p.pdf>
- BEA Systems. (2003). On XML optimization. Retrieved October 20, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/26-bea-BinaryXMLWS.pdf>
- Beard, A. (April 2007). A Survey on open source software licenses. *Journal of Computing Sciences in Colleges*, 22(4), 205–211.
- Blanc, B. & Maaraoui, B. (2005, December) White paper: Endianness or where is byte 0. Retrieved September 10, 2009 from 3B-Consultanc Web site: <http://3bc.bertrand-blanc.com/endianness05.pdf>
- Bradski, G. & Kaehler, A. (2008, September). *Learning OpenCV Computer Vision with the OpenCV Library*. Sebastopol, CA: O'Reilly.
- Brauer, M. & Schubert, S. (n.d.). The OpenOffice.org XML Project. Retrieved December 18, 2008, from OpenOffice Web site: <http://xml.openoffice.org/>

- Brutzman, D. & McGregor, D. (2003, September 28). XML binary serialization using cross-format schema protocol (XFSP) and XML compression considerations for extensible 3D (X3D) graphics. Retrieved November 6, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/40-BrutzmanXmlBinarySerializationUsingXfspW3cWorkshopSeptember2003.pdf>
- Brutzman, D. & Daly, L. (2007). *X3D: Extensible 3D graphics for Web authors*. San Francisco, CA: Morgan Kaufmann.
- Brutzman, D. & Blais, C. (2010, February 22). Track data conversion suite: building track interoperability for ASW COI [PowerPoint slides]. Retrieved September 4, 2009, from Naval Postgraduate School, Modeling, Virtual Environments and Simulation Institute available by request from Brutzman@nps.edu or clblais@nps.edu.
- Bush, G. (2004, August 27). Executive Order 13356. Retrieved December 28, 2008 from Federation of American Scientist Web site: <http://www.fas.org/irp/offdocs/eo/eo-13356.htm>
- Buss, A. (2001, November). TECHNICAL NOTES Discrete Event Programming with Simkit. *Simulation News Europe*, 32/33, 15–25.
- Buss, A. (2002). Component Based Simulation Modeling With Simkit. *Proceedings of the 2002 Winter Simulation Conference*, 12, 243–249.
- Buss, A., (2009). Discrete Event Simulation Modeling (OA3302). Monterey, CA: MOVES Institute. Available upon request abuss@nps.edu.
- Cannon Information Systems Research. (2003). Position paper on the W3C XML binary interchange. Retrieved from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/18-canon-au-xml-binary.html>
- Carey, P. (2007). *New Perspectives on XML 2nd Edition – Comprehensive*. Boston, MA: Thomson.
- Childers, C.M. Applying semantic Web concepts to support net-centric warfare using the tactical assessment markup language (TAML). Master's thesis, Department of Computer Science, Naval Postgraduate School, Monterey, California, June 2006.

- Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, 113–124. Retrieved from September 2, 2009, the Noam Chomsky Web site: <http://www.chomsky.info/articles/195609--.pdf>.
- Cisco, Inc. (2003). XML processing in the network Cisco position paper to the W3C workshop on binary interchange of XML information item sets. Retrieved December 5, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/41-CiscoW3CBinaryXMLWorkshopPositionPaper.htm>
- CNO (2005, April). Navy Operational Designated Approving Authority (DAA) Responsibilities and Authority, United States Naval Message, 152315Z APR 05.
- Computer Engineering and Networks Laboratory. (2003, August). Position paper for the W3C workshop on binary interchange of XML information item sets. Retrieved September 9, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/23-wilde-w3c-bxml.pdf>
- CSC/NASA. (2003). Binary representation of XML infoset in the space domain position paper. Retrieved September 15, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/14-ccds-w3cposition-updated.pdf>
- Cube Werx inc. (2003). CubeWerx position paper for binary interchange of XML. Retrieved November 3, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/05-cubewerx-position-w3c-bxml.pdf>
- DA&M. (2006, July). Defense Information System Agency (DISA), DoDD 5105.19. Retrieved December 5, 2009, from <http://www.dtic.mil/whs/directives/corres/pdf/510519p.pdf>
- Davis, D.T. Design, implementation and testing of a common data model supporting autonomous vehicle compatibility and interoperability. Dissertation, Department of Computer Science, Naval Postgraduate School, Monterey, California, September 2006.
- Davis, E.L. Evaluation of the extensible markup language (XML) as a means of establishing interoperability between multiple department of defense (DoD) databases. Master's thesis, Department of Software Engineering, Naval Postgraduate School, Monterey, California, June 2001.

- Davis, M. E. & Weyuker, E. J. (1994). *Computability, complexity, and languages: Fundamentals of theoretical computer science*. Boston, MA: Academic Press.
- Day, Ed. (2007, December 02). W3C efficient XML interchange (EXI). Retrieved September 9, 2009, from Objective Systems Web site: <http://www.objsys.com/binxml/EXILightning.pdf>
- Delta3D. (n.d.). Delta3D open source gaming and simulation engine. Retrieved November 2, 2009, from <http://www.delta3d.org/>
- Denny, I.M. & Jahn, D. Performance comparison of relational and native-XML databases using the semantics of the land command and control information exchange data model (LC2IEDM). Master's thesis, Department of Information Science, Naval Postgraduate School, Monterey, California, September 2005.
- DeVore, J. (2008). *Probability and statistics for engineering and the sciences*. Belmont, CA: Thomson.
- DeVos, D.A. XML Tactical Chat (XTC): The way ahead for Navy chat. Master's thesis, Department of Information Science, Naval Postgraduate School, Monterey, California, September 2007.
- DIS. (1995, September). *IEEE standard for distributed interactive simulations application protocol*. New York, NY: IEEE.
- DISA. (n.d.) Defense information system agency homepage. Retrieved May 30, 2009, from <http://www.disa.mil/>
- DoD CIO. (2001, April 6) DoD Chief Information Officer (CIO) Guidance and Policy Memorandum (G&PM) No. 11-8450, Department of Defense (DoD) Global Information Grid (GIG) computing. Retrieved September 9, 2009, from DTIC online: Information for the Defense Community Web site: <http://www.dtic.mil/whs/directives/corres/pdf/dsd010406gig.pdf>
- DoD CIO. (2003, May 9). Department of Defense net-centric data strategy. Retrieved September 12, 2009, from <http://www.dod.mil/cio-nii/docs/Net-Centric-Data-Strategy-2003-05-092.pdf>
- DoD CIO. (2004, May 12). Net-centric checklist version 2.1.3. Retrieved September 7, 2009, from http://www.dod.mil/cio-nii/docs/NetCentric_Checklist_v2-1-3_.pdf

- DoD CIO IM. (2006, March 13). Net centric operations conference. Transforming the Way DoD Manages Data [PowerPoint slides]. Retrieved September 9, 2009, from <http://www.dtic.mil/ndia/2006netcentric/risacher.pdf>
- DoD CIO. (2007, April 23) Information assurance, DoDD 8500.01E. Retrieved November 12, 2009, from <http://www.dtic.mil/whs/directives/corres/pdf/850001p.pdf>
- DoD CIO. (2007, May 4). Department of Defense net-centric services strategy: Strategy for a net-centric, service orientated DoD enterprise. Retrieved November 12, 2009, from http://www.defenselink.mil/cio-nii/docs/Services_Strategy.pdf
- DoD CIO. (2007, November) DoD information assurance certification and accreditation process (DIACAP), DoDI 8510.01. Retrieved September 12, 2009, from <http://www.dtic.mil/whs/directives/corres/pdf/851001p.pdf>
- DoD CIO. (2008, December) NetOps for the global information grid (GIG), DoDI 8410.02. Retrieved September 12, 2009, from <http://www.dtic.mil/whs/directives/corres/pdf/841002p.pdf>
- DON CIO. (2002, December 13). DON policy on the use of extensible markup language (XML). Retrieved September 12, 2009, from Cover pages Web site: <http://xml.coverpages.org/DON-XMLPolicy200212.pdf>
- DON CIO. (2005, January). Department of the Navy XML naming and design rules. Retrieved September 9, 2009, from XML.gov Web site: <http://www.xml.gov/documents/completed/don/ndr2.pdf>
- DON CIO. (2005, June 06). Renewal of the charter for the department of the Navy extensible markup language business standards council. Retrieved September 29, 2009, from <http://www.doncio.navy.mil/EATool/Documents/BSCCharterJune2005.pdf>
- DON CIO. (2008, May 13). Department of the Navy Naval Networking Environment (NNE)~2016: Strategic Definition, Scope and Strategy Paper. Retrieved from <http://www.doncio.navy.mil/Download.aspx?AttachID=565>
- Estlund, M.J. A survey and analysis of access control architectures for XML data. Master's thesis, Department of Computer Science, Naval Postgraduate School, Monterey, California, March 2006.

- Expway. (2003, August 07). Expway's position paper on binary infosets. Retrieved September 9, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/15-Expway-PositionPaper-20031020.zip>
- Filiagos, D.E. Developing an after action review system for a 3D interactive training simulation using XML. Master's thesis, Department of Modeling, Virtual Environments and Simulation (MOVES), Naval Postgraduate School, Monterey, California, March 2004.
- FreeBSD. (n.d.) The FreeBSD Project. Retrieved May 3, 2009, from <http://www.freebsd.org/>
- GNU. (2009, April). The GNU operating system. Retrieved April 28, 2009, from <http://www.gnu.org/>
- GZip. (2003, July 27). The GZip homepage. Retrieved September 19, 2009 from www.gzip.org
- Hand, D., Mannila, H., & Smyth, P. (2001, August). *Principles of data mining*. Cambridge: Massachusetts Institute of Technology
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. New York: Springer Science.
- Hina, D.R. Evaluation of the extensible markup language (XML) as a means of establishing interoperability between heterogeneous department of defense (DoD) databases. Master's thesis, Department of Software Engineering, Naval Postgraduate School, Monterey, California, September 2000.
- Hodges, G.A. Designing a common interchange format for unit data using the command and control information exchange data model (C2IEDM) and XSLT. Master's thesis, Department of Modeling, Virtual Environments and Simulation (MOVES), Naval Postgraduate School, Monterey, California, September 2004.
- Hout, G.K. Toward XML representation of NSS simulation scenario for mission scenario exchange capability. Master's thesis, Department of Modeling, Virtual Environments and Simulation (MOVES), Naval Postgraduate School, Monterey, California, September 2003.
- Hunter D., Cagle, K., Dix, C., Kovack, R., Pinnock, J., & Rafter, J. (2001) *Beginning XML*. 2nd Edition. Birmingham, UK: Wrox.

- HiT Software, Inc. (2003). Compressing and Filtering XML Streams. Retrieved September 12, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/33-HiT-W3C-Workshop-2003.pdf>
- IBM Corporation. (2003). Issues relating to the creation of a binary interchange standard for XML. Retrieved September 12, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/19-IBM-IBMPositionPaperBinaryXMLWorkshop-updated.html>
- IETF. (1999). Hypertext transfer protocol -- HTTP/1.1 RFC 2616. Retrieved September 9, 2009, from <http://www.ietf.org/rfc/rfc2616.txt>
- Inkscape. (n.d.). Inkscape: Open source scalable vector graphics editor. Retrieved November 12, 2009, from <http://www.inkscape.org/>
- Jabber. (2006, August 14). XMPP Emerging as Chat Standard for the Federal Government. Retrieved December 29, 2009 from <http://www.webwire.com/ViewPressRel.asp?aId=18455>
- Jacobs, M., (2008, November 23). DON XML –Achieving Enterprise Interoperability [Power Point Slides]. Retrieved September 9, 2009, from National Defense Industrial Associates Web site: http://proceedings.ndia.org/3690/Tuesday_Breakout_RoomB/DON_CIO.pdf
- Japex. (n.d.) Japex: Japex Micro-benchmark Framework. Retrieved March 16, 2010 from <https://japex.dev.java.net/>
- JUnit. (4 August 2009). Java Unit Testing Resource 4.7. Retrieved November 22, 2009, from SourceForge: Find and Develop Open source Software Web site: <http://sourceforge.net/projects/junit/files/junit/>
- Kane, D.R.Jr. Web-based dissemination system for the trusted computing exemplar project. Master's thesis, Department of Computer Science, Naval Postgraduate School, Monterey, California, June 2005.
- Kayne, R. (2010, January 19). What is software licensing? Retrieved January 22, 2010 from WiseGeek: Clear Answers for Common Questions Web site <http://www.wisegeek.com/what-is-software-licensing.htm>

- KDDI R&D Labs. (2003). Implementation and evaluation of a binary interchange system for XML-applications in a cellular phone. Retrieved September 22, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/34-KDDI-Binary-XML.pdf>
- Keynote. (1999). The keynote trust-management system Version 2, RFC 2704. Retrieved December 5, 2009, from <http://www.cis.upenn.edu/~angelos/Papers/rfc2704.txt>
- L3 Communications Integrated Systems. (2003, August 11). The L-3 communications, integrated systems position regarding binary interchange of XML information. Retrieved from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: http://www.w3.org/2003/08/binary-interchange-workshop/23a-L3IS_BinaryXML_Position_11Aug03.pdf
- Lionet. (2003, August 11). Binary XML transfer using direct compilation techniques. Retrieved September 22, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/12-lionet-HornXMLTransfer.pdf>
- Lai, E. (2008, March 17). Social-networking site keeps Apache atop Web server market. Retrieved 15 January 2009, from Computerworld The Voice of IT Management Web site: <http://www.computerworld.com.au/index.php/id;1569609646;pp;2;fp;;fpid;>
- Langevin, J.R., McCaul, M. T., Charney, S. & Raduege, H. (2008, December). Securing Cyberspace for the 44th Presidency: A Report of the CSIS Commission on Cybersecurtiy for the 44th Presidency. Center for Strategic and International Studies. Retrieved September 22, 2009, from http://csis.org/files/media/csis/pubs/081208_securingcyberspace_44.pdf
- Laurent, A. M. (2004) *Understanding open source and free software licensing, guide to navigating licensing issues in existing & new software*. Sebastopol, CA: O'Reilly Media.
- Lawler, G.M. Distributed architecture for the object-oriented method for interoperability. Master's thesis, Department of Computer Science, Naval Postgraduate School, Monterey, California, March 2003.
- Lee, T.B. (1999) *Weaving the Web: The original design and ultimate destiny of the World Wide Web*. New York, NY: Harper Collins.

- License. (2009). In *Merriam-Webster Online Dictionary*. Retrieved August 28, 2009, from <http://www.merriam-webster.com/dictionary/License>
- Liefke, H. & Suciu D. (2000). XMill: an efficient compressor for XML data. In *Proceedings of SIGMOD*, 153–164. Retrieved from <http://www.cs.washington.edu/homes/suciu/xmill.ps>
- Marshall, R.(2003, August 11). Binary representation of XML, a position. Retrieved from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/13-XML-Binary-Representation.ps>
- McCarty, G.E. Jr. Integrating XML and RDF concepts to achieve automation within a tactical knowledge management environment. Master's thesis, Department of Software Engineering, Naval Postgraduate School, Monterey, California, March 2004.
- Media Fusion Corporation. (2003). Serializing DOM method position paper for the W3C workshop on binary interchange of XML information item sets. Retrieved September 22, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: http://www.w3.org/2003/08/binary-interchange-workshop/21-PositionPaper_MediaFusion.zip
- Media Fusion. (n.d.) Media fusion XML storage solutions. Retrieved March 12, 2009, from <http://www.mediafusion.co.jp/company/obj.html>
- Michaelson, J. (2004, May) There's nNo such thing as a free (software) lunch. *Queue*. 2(3), 40–47.
- Microsoft. (2006, June). Walkthrough: Word 2007 XML format. Retrieved July 15, 2009 from <http://msdn.microsoft.com/en-us/library/bb266220.aspx>
- Microsoft Corporation. (2003). A case against standardizing binary representation of XML. Retrieved November 18, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/29-MicrosoftPosition.htm>
- MITRE Corporation. (2008, May). Analysis of the XML size reduction using the efficient XML interchange (Report MTR080127). MITRE.

- MITRE Corporation. (2003). Binary XML position paper: The need for standard schema-based and hybrid compression. Retrieved September 2, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/25-MITRE-USA-Binary-XML.htm>
- Mohan, R. XML based adaptive IPSEC policy management in a trust management context. Master's thesis, Department of Computer Science, Naval Postgraduate School, Monterey, California, September 2002.
- MSDN. (2008, December 18). Open XML formats resource center. Retrieved July 17, 2009 from <http://msdn.microsoft.com/en-us/office/bb265236.aspx>
- OpenBSD. (2009, December 1). OpenBSD free, functional and secure. Retrieved January 22, 2010 from <http://www.openbsd.org/>
- Netcraft, (2009, April). April 2009 Web Server Survey. Retrieved April 29, 2009 from http://news.netcraft.com/archives/2009/04/06/april_2009_web_server_survey.html
- Neushul, J.D. Interoperability, data control and battlespace visualization using XML, XSLT and X3D. Master's thesis, Department of Computer Science, Naval Postgraduate School, Monterey, California, September 2003.
- Nokia. (2003). Nokia position paper: W3C workshop on binary interchange of XML information item sets. Retrieved December 5, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: http://www.w3.org/2003/08/binary-interchange-workshop/02-Nokia-Position-Paper_02.htm
- Norbraten, T.D. Utilization of forward error correction (FEC) techniques with extensible markup language (XML) schema-based binary compression (XSBC) technology. Master's thesis, Department of Modeling, Virtual Environments and Simulation (MOVES), Naval Postgraduate School, Monterey, California, December 2004.
- Nordquist, P., Petersen, A., & Todorova, A. (2003, December). License tracing in free, open, and proprietary software. *Journal of Circuits, Systems and Computers*, 19(2), 101–112.
- Ontonet. (2003). Position paper for the W3C workshop on binary interchange of XML information item sets. Retrieved September 12, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/24-ontonet-BinaryInfosetPositionPaper.html>

- Open-DIS, (n.d.), Open-DIS. Retrieved July 19, 2009, from SourceForge: Find and Develop Open source Software Web site: <http://open-dis.sourceforge.net>
- Oracle Corporation. (2003). Position paper for the W3C workshop on binary interchange of XML information item sets, Oracle Corporation. Retrieved October 22, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: http://www.w3.org/2003/08/binary-interchange-workshop/31-oracle-BinaryXML_pos.htm
- OSS Nokalva Inc. (2003, August 11). Alternative binary representations of the XML information set based on ASN.1. Retrieved September 17, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/32-OSS-Nokalva-Position-Paper-updated.pdf>
- PKware, (2007, September 28). APPNOTE.TXT - .ZIP file format specification version 6.3.2. Retrieved December 8, 2009, from <http://www.pkware.com/documents/casestudies/APPNOTE.TXT>
- PKware. (n.d.). PKWARE data security & compression software. Retrieved January 10, 2010, from <http://www.pkware.com/>
- Popovich R.M. (2005, October). *Information and communications managers course student handbook (A-202-0041)*. San Diego, CA: MR Popovich and Company.
- Pradeep, K. XML as a data exchange medium for DoD legacy databases, Master's thesis, Department of Software Engineering, Naval Postgraduate School, Monterey, California, June 2002.
- Prague, C. N., Irwin, M. R., & Reardon, J. (2003). *Microsoft office access 2003 bible*. Indianapolis, IN: Wiley.
- Reynolds, L. K. A framework for the management of evolving requirements in software systems supporting network-centric warfare, Master's thesis, Department of Computer Science, Naval Postgraduate School, Monterey, California, June 2006.
- Rosetti, S. Tactical Web services: Using XML and Java Web services to conduct real-time net-centric sonar visualization. Master's thesis, Department of Modeling, Virtual Environments and Simulation (MOVES), Naval Postgraduate School, Monterey, California, September 2004.
- Ross, S. M. (2007). *Introduction to probability models*. Oxford, UK: Elsevier.

- Sanchez, P., (2009). Simulation analysis (Fall 2009 OA4333). Monterey, CA: Simulation, Experiments and Efficient Design (SEED) Center. Available upon request pjsanche@nps.edu or <http://or.nps.edu/faculty/PaulSanchez/oa4333/> and <http://harvest.nps.edu/>
- SAP. (2003). W3C workshop on binary interchange of XML information item sets. SAP Position Paper. Retrieved September 10, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/21a-W3CWorkshop-SAPPositionPaper.pdf>
- Siemens. (2003, August 08). Reply to call for participation in W3C workshop on binary interchange of XML information item sets. Retrieved November 1, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: http://www.w3.org/2003/08/binary-interchange-workshop/39-siemens-Brief_W3C_Workshop_030809_1.pdf
- Siemens. (n.d.). <EXIficient/> XML becomes efficient. Retrieved July 19, 2009, from SourceForge: Find and Develop Open source Software Web site: <http://exificient.sourceforge.net>
- SIMKIT. (2010, January 4). SIMKIT home page. Retrieved January 12, 2010 from <http://diana.cs.navy.mil/simkit/>
- Software AG. (2003, August 08). Software AG position on binary XML. Retrieved September 22, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/17-softwareAG-BinaryPosition.html>
- Sosnoski Software Solutions inc. (2003). XBIS XML info set encoding. Retrieved November 12, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/09-Sosnoski-position-paper.pdf>
- Stern, N. & Stern, R. (1994, December). *Structured COBOL programming, 8th Edition with syntax guide and student program*. Sebastopol, CA: John Wiley.
- Stewart, J. D. An XML-based knowledge management system of port information for U.S. Coast Guard cutters. Master's thesis, Department of Information Science, Naval Postgraduate School, Monterey, California, March 2003.

- Sun Microsystems. (2003, July 24). Fast Web services. Retrieved September 3, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets website: http://www.w3.org/2003/08/binary-interchange-workshop/01-FWS_Sun.pdf
- Sun Microsystems. (n.d.). jar-The Java archive tool. Retrieved September 19, 2009, from <http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/jar.html>
- Systematic Software Engineering. (2003, August 11). The W3C workshop on binary interchange of XML information Item sets position paper systematic software engineering, [Online]. Retrieved September 3, 2009 from <http://www.w3.org/2003/08/binary-interchange-workshop/22-SSE-0001W3CPositionPaper.pdf>
- Tar. (2009, March 05). Tar – GNU project – free software foundation (FSF). Retrieved November 17, 2009, from <http://www.gnu.org/software/tar/>
- Tarari. (2003). XML binary infosets: Position paper from tarari. Retrieved from October 2, 2009, The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/27-tarari-BinaryInterchangeOfXML.pdf>
- TeliaSonera. (2003, August 07). Position paper – Binary interchange of XML information item sets. Retrieved September 17, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: http://www.w3.org/2003/08/binary-interchange-workshop/07-TeliaSonera_Position_Paper_07082003.pdf
- Timed Text Working Group. (2003). Position paper on binary interchange of XML W3C timed text working group. Retrieved September 14, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchange-workshop/03-GlennAdams.txt>
- Ultra Life Batteries. (n.d.). Battery technology and Moore's Law. Retrieved December 17, 2008, from http://www.ultralifebatteries.com/battery_trends.php?ID=6
- Unicode (2009, December 23). Unicode 5.0.2. Retrieved January 7, 2010, from <http://www.unicode.org/versions/Unicode5.2.0/>
- University of Helsinki. (2003). Byte-efficient representation of XML messages. Retrieved from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: <http://www.w3.org/2003/08/binary-interchangeworkshop/08-xebu.pdf>

- USD. (2003, December 22). Migration to the defense logistics management standards (DLMS) and elimination of the military standards system (MILS). Retrieved December 5, 2009, from Defense Logistics Agency Web site:
<http://www.dla.mil/j-6/dlms/Programs/DLMS/endDLSSmemo.pdf>
- VISKIT. (2008, August 8). VISKIT Home Page. Retrieved July 19, 2009, from
<http://diana.cs.nps.navy.mil/Viskit/>
- W3C. (1999, March 12). XML in 10 Points. Retrieved February 18, 2009, from
<http://www.w3.org/XML/1999/XML-in-10-points>
- W3C. (2005, January 25). SOAP message transmission optimization mechanism. Retrieved September 2, 2009, from <http://www.w3.org/TR/soap12-mtom/>
- W3C. (2005, January 25). XML-binary optimized packaging. Retrieved November 3, 2009, from <http://www.w3.org/TR/2005/REC-xop10-20050125>
- W3C. (2005, March 31). XML binary characterization use cases. Retrieved September 12, 2009 from <http://www.w3.org/TR/2005/NOTE-xbc-use-cases-20050331>
- W3C. (2005, October 14). World Wide Web Consortium process document (Chapter 7). Retrieved December 2, 2009, from <http://www.w3.org/2005/10/Process-20051014/cover.html#toc>
- W3C. (2006, July 18). Efficient XML interchange measurements note retrieved September 23, 2009, from <http://www.w3.org/TR/2006/WD-exi-measurements-20060718/>
- W3C. (2007, July 25). Efficient XML interchange measurements note. Retrieved October 7, 2009, from <http://www.w3.org/TR/2007/WD-exi-measurements-20070725>
- W3C. (2007, December 19). Efficient XML interchange (EXI) primer. Retrieved September 9, 2009, from <http://www.w3.org/TR/2007/WD-exi-primer-20071219>
- W3C. (2007, December 19). Efficient XML interchange (EXI) best practices. Retrieved September 9, 2009, from <http://www.w3.org/TR/2007/WD-exi-best-practices-20071219>
- W3C. (2008, April 10). EXI 1.0 encoding examples. Retrieved September 9, 2009, from
<http://www.w3.org/XML/EXI/tutorial/exi-examples.html>

- W3C. (2008, July 28). Efficient XML interchange evaluation. Retrieved September 9, 2009, from <http://www.w3.org/TR/2008/WD-exi-evaluation-20080728>
- W3C. (2008, September 03). Efficient XML interchange (EXI) impacts. Retrieved September 15, 2009, from <http://www.w3.org/TR/2008/WD-exi-impacts-20080903>
- W3C. (2008, September 19). Efficient XML interchange (EXI) format 1.0. Retrieved September 15, 2009, from <http://www.w3.org/TR/2008/WD-exi-20080919>
- W3C, (2010, March 7). Resource Description Framework (RDF). Retrieved March 15, 2010, 2010 from <http://www.w3.org/RDF/>
- W3C. (n.d.). World Wide Web Consortium (W3C) homepage. Retrieved September 19, 2009, from <http://www.w3.org/>
- Wakenbach, J. (2003). *Microsoft Office Excel 2003 bible*. Indianapolis, IN: Wiley.
- WHS, (2008, April 18). Office of the Secretary of Defense (OSD) Records Management. Retrieved September 30, 2009, from <http://www.dtic.mil/whs/directives/corres/pdf/a015v1p.pdf>
- WHS. (2010, January 28). Washington Headquarters Service. Retrieved January 28, 2010, from <http://www.whs.mil/About/WHSHistory.cfm>
- Williams, J. S. Document-centric XML encryption and authentication for coalition messaging. Master's thesis, Naval Postgraduate School, Monterey California, September 2009.
- Williams, S. (2002, March). *Free as in freedom: Richard Stallman's crusade for free software*. Sebastopol, CA: O'Reilly.
- Williams, S. D. (2003, August 08). Position paper for the W3C workshop on binary interchange of XML information item sets. Retrieved December 15, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web site: http://www.w3.org/2003/08/binary-interchange-workshop/10-w3cbisposition_sdw.html
- Williams, S, D. (2009, February 16). Stephen D. Williams says. Retrieved March 12, 2009, from <http://sdw.st/>

XimpleWare. (2003, August 10). Ximpleware W3C position paper. Retrieved September 9, 2009, from The W3C Workshop on Binary Interchange of XML Information Item Sets Web –8e: <http://www.w3.org/2003/08/binary-interchange-workshop/20-ximpleware-positionpaper-updated.htm>

Zip. (2010, January 24). ZIP (File format) Wikipedia, the free encyclopedia. Retrieved January 28, 2010, from [http://en.wikipedia.org/wiki/ZIP_\(file_format\)](http://en.wikipedia.org/wiki/ZIP_(file_format))

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, VA
2. Dudley Knox Library
Naval Postgraduate School
Monterey, CA
3. Don Brutzman, Ph.D
Naval Postgraduate School
Monterey, CA
4. Don McGregor
Naval Postgraduate School
Monterey, CA
5. Mr. Robert J. Carey
Department of the Navy Chief Information Officer
1000 Navy Pentagon
Washington, DC
6. Mr. David M. Wennergren
Deputy Assistant Secretary of Defense for Information
Management and Technology & DoD Deputy Chief Information Officer
1400 Defense Pentagon
Washington DC